明 細 書

ソフトウェア生成方法

技術分野

[0001] 本発明は、ソフトウェア生成方法に係り、特に、Lyee方法論におけるプロセス代数による形式化を用いたソフトウェア生成方法に関する。

背景技術

[0002] ここ数年の間に、ソフトウェアの開発ライフサイクルに関連する1つまたは多くの局面を改善するために種々の方法論および技術が考案され提案されてきた。しかし、この研究分野における熱心な努力にもかかわらず、明快に理解でき、修正可能なシステムの製造は、いまだに野心的目標であり達成するにはほど遠い。その理由の1つは、ソフトウェア自身が複雑なものであり、捕らえにくいものであるからであり、もう1つの理由は、現在の方法論に限界があるからである。最近、Lyee(商標)と呼ばれる新しく、非常に有望な方法論が提案された。異なる分野に関連する広い範囲のソフトウェアの問題を効率的に処理することを目的としたLyeeにより、その要件を定義するだけでソフトウェアを開発することができるようになった。

[0003] しかし、Lyeeにより生成されたソフトウェアの意味論も、要件からのソフトウェア自動生成プロセスも、形式的でない言語で説明されるので、この方法論を理解し研究しようとすると、難しく、混乱が生じる恐れがある。

非特許文献1:ジェイ・エイ・ベルグストゥラ、ジェイ・ダブリュ・クロップ(J.A. Bergstra and J. W. Klop)著、「抽象化を用いたコミュニケーション・プロセスの代数」、1985年, 理論コンピュータ科学、37(1)、p. 77−121

非特許文献2:ジー・ ベリー、ジー・バウドル(G. Berry and G. Boudol)著、「化学抽象機械」、1992年, 理論コンピュータ科学、96(1)、p. 217−248

非特許文献3:シー・エー・アール・ホーア(C.A.R.Hoare)著、「コミュニケーティング順次プロセス」、プレンティス・ホールコンピュータ科学国際シリーズ、プレンティス・ホール出版、1985年

非特許文献4:エム・メジリ、ビー・クタリ、エム・アーヒオウイ(M.Mejri, B.Ktari, and

M.Erhioui)著、「Lyee指向ソフトウェアについての静的分析」、パリにおけるLyee方法論に関する第1回国際ワークショップ予稿集、ハミド・藤田、ポウル・ジョナネッセン編「ソフトウェア方法論、ツール及び技術における新潮流」p. 375−394、アイオーエス出版、2002年

非特許文献5:アール・ミルナー(R.Milner)著、「コミュニケーション・システムの計算法」、コンピュータ科学92講義録、ベルリン、1980年、スプリンガー・フェアラーク出版

非特許文献6:根来文生著、「Lyeeソフトウェアの原理」、21世紀の情報科学についての国際会議2000(IS2000)、p. 121−189、2000年11月

非特許文献7:根来文生著、「Lyee入門」、ソフトウェア生産技術研究所、東京、日本、2001年

非特許文献8:根来文生、アイ・ハミド(I.Hamid)著、「意思工学の提案」、データベースと情報システムに関する第5回東ヨーロッパ会議研究 (ADBIS'2001)、2000年9月

非特許文献9:根来文生、アイ・ハミド(I.Hamid)著、「意思工学の提案」、インターネット上の電子商取引、科学及び教育の基盤における進歩に関する国際会議 (SSGRR2001)、2001年

発明の開示

発明が解決しようとする課題

[0004] 本発明の主な目的は、第1に、プロセス代数を用いて、Lyeeにより生成したソフトウェアの意味論とソフトウェアの自動生成プロセスを形式化することである。実際、プロセス代数は、本質的にLyee方法論の多くの概念を裏付けるので、従ってLyee方法論を簡潔かつ巧みに形式化する。そして、第2の目的は、プロセス代数は、フォン・ノイマンのものより、より適した抽象機械をLyee方法論に提供することである。実際、この新しい抽象機械は、プログラムを化学溶液と見なし、そこでは、分子(Lyee方法論の異なるベクトル)が共通の目的を達成するために相互に作用しあっている。

課題を解決するための手段

[0005] かかる課題を解決するために本発明は、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとに、単語名、定義式、該定義式の実行

条件、入出力属性、単語の値の属性によって宣言(規定)する第1のステップと、単語単位の宣言から、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)を作成する第2のステップと、前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合に集合化する第3のステップと、前記集合ごとに、1つの制御関数モジュールΦを配置する第4のステップと、前記プログラムに1つの制御関数モジュールΨを配置する第5のステップとを具備する。

[0006] 本発明の異なる実施体としての「開発対象のソフトウェア」を生産するためのプログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開発支援装置、ソフトウェア開発管理装置は、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型の未定義部分に、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を挿入する、宣言情報挿入手段と、前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦを、前記単位を制御するように関連づける、制御関数Φ配置手段と、1つの制御関数モジュールΨを、前記制御関数Φを制御するように前記制御関数Φに関連づける、制御関数Ψ配置手段とを具備するように構成することもできる。

[0007] 本発明はさらに、上述の「開発対象のソフトウェアを生産する方法」によって生産されたソフトウェア、及び当該ソフトウェアが搭載された記録媒体或いは当該ソフトウェアが搭載された装置(ハードウェア)としても実現されるが、この場合の本発明は、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型の未定義部分に挿入したモジュール群と、前記モジュール群を、同一画面

からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦで制御するように関連づけている1つまたは複数の制御関数モジュールΦと、前記制御関数Φを、1つの制御関数モジュールΨで制御するように関連づけている制御関数モジュールΨで構成されることもできる。

[0008] またさらに本発明は、上述の「開発対象のソフトウェアを生産する方法」によってソフトウェアを生産するために用いられるソフトウェアコードの雛型としてのソフトウェア、及び当該ソフトウェアが搭載された記録媒体或いは当該ソフトウェアが搭載された装置(ハードウェア)としても実現されるが、この場合の本発明は、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を、埋め込むべき未定義部分を有した、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型と、前記宣言の情報を未定義部分に挿入した前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦで制御するための機能を有した制御関数モジュールΦのひな型と、前記制御関数Φを1つの制御関数モジュールΨで制御するための機能を有した制御関数モジュールΨのひな型とを備えるソフトウェアとしてコード化可能なひな型として実現してもよい。

[0009] さらに、本発明は、上述の「開発対象のソフトウェアを生産する方法」による、要件から抽出した情報(ドキュメント(紙、データ))の抽出方法として、またかかる抽出方法によって抽出された情報(ドキュメント(紙、データ))として、さらには当該抽出された情報の使用方法として、或いは、これらの情報が搭載された情報記録媒体として、または情報の抽出方法／使用方法がコード化されたソフトウェア、当該ソフトウェアが搭載された記録媒体／装置(ハードウェア)として、いずれも実現することができるが、この場合の本発明は、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型の未定義部分に挿入すべき、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出

力属性、単語の値の属性からなる宣言として情報化した情報と、前記宣言の情報を未定義部分に挿入した前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦで制御するように関連づけるための情報と、前記制御関数Φを1つの制御関数モジュールΨで制御するように関連づける情報とを備えるソフトウェア開発要件から抽出した情報として実現してもよい。

[0010] なお、論理体については、同一出願人による特願2004−272400を参照し、これを引用し、開示の一部とする。

発明の効果

[0011] 異なる分野に関連する広範囲のソフトウェアの問題を効率的に処理し、従来の方法論と比較した場合には、Lyeeを使用すると開発時間、保守時間およびドキュメント量はかなり少なくなる(70ー80%程度)。

発明を実施するための最良の形態

[0012] Lyee方法論:プロセス代数による形式化

概要:ここ数年の間に、ソフトウェアの開発ライフサイクルに関連する1つまたは多くの局面を改善するために種々の方法論および技術が考案され提案されてきた。しかし、この研究分野における熱心な努力にもかかわらず、明快に理解でき、修正可能なシステムの製造は、いまだに野心的目標であり達成するにはほど遠い。その理由の1つは、ソフトウェア自身が複雑なものであり、捕らえにくいものであるからであり、もう1つの理由は、現在の方法論に限界があるからである。最近、Lyeeと呼ばれる新しく、非常に有望な方法論が提案された。異なる分野に関連する広い範囲のソフトウェアの問題を効率的に処理することを目的としたLyeeにより、その要件を定義するだけでソフトウェアを開発することができるようになった。

[0013] しかし、Lyeeにより生成されたソフトウェアの意味論も、要件からのソフトウェア自動生成プロセスも、形式的でない言語で説明されるので、この方法論を理解し研究しようとすると、難しく、混乱が生じる恐れがある。

[0014] 本発明の主な目的は、第1に、プロセス代数を用いて、Lyeeにより生成したソフトウェアの意味論とソフトウェアの自動生成プロセスを形式化することである。実際、プロ

セス代数は、本質的にLyee方法論の多くの概念を裏付けるので、従ってLyee方法論を簡潔かつ巧みに形式化する。そして、第2の目的は、プロセス代数は、フォン・ノイマンのものより、より適した抽象機械をLyee方法論に提供することである。実際、この新しい抽象機械は、プログラムを化学溶液と見なし、そこでは、分子(Lyee方法論の異なるベクトル)が共通の目的を達成するために相互に作用しあっている。

1. 始めに

高い品質を持つソフトウェアを容易に迅速に製造することは、ソフトウェア開発研究分野の基本的な関心事である。ここ数年にわたって、ソフトウェアの開発ライフサイクルに関連する1つまたは多くの局面を改善するために、種々の方法論および技術が考案され提案されてきた。しかし、この研究分野における熱心な努力にもかかわらず、明快に理解でき、修正可能なシステムの製造は、いまだに野心的目標であり達成するにはほど遠い。その理由の1つは、ソフトウェア自身が複雑なものであり、捕らえにくいものであるからであり、もう1つの理由は、現在の方法論に限界があるからである。実際、提案されたほとんどすべての方法論は、明快に理解でき、修正可能なシステムの製造に失敗しており、その使用は、いまだに非常に広範囲の能力、技術および知識を持つ専門家だけが可能なことだと見なされている。そのため、人件費や維持費が高いものになり、ソフトウェアに対して広範なチェックが必要になる。これらの理由で、企業は、ソフトウェア開発サイクルにおける実証可能な改善を約束するすべての新しい方法論を歓迎する傾向にある。

[0015] 最近、Lyee(「リー」と読む。「governementaL methodologY for softwarE providencE」の尾文字語)(非特許文献6、7、8、9参照)と呼ばれる新しく、非常に有望な方法論が提案された。異なる分野に関連する広範囲のソフトウェアの問題を効率的に処理することを目的としたLyeeにより、その要件を定義するだけでソフトウェアを開発することができる。もっと正確に言えば、開発者は、単語、計算式、計算条件(前提条件)および画面と帳票のレイアウトを与えさえすれば、後は、コンピュータが、以降のすべての面倒なプログラミング・プロセス(例えば、制御ロジックに関する局面など)をすべて行ってくれる。その新しさにもかかわらず、Lyeeの使用結果は、Lyee

には非常に多くの潜在的可能性があることを示している。実際、従来の方法論と比較した場合には、Lyeeを使用すると開発時間、保守時間およびドキュメント量はかなり少なくなる(70ー80%程度)(非特許文献7参照)。非特許文献4において、この方法論の多くの、特にLyee要件の、局面を改善するために、我々は、いくつかの古典的静的分析技術を提案した。

[0016] しかし、Lyeeにより生成したソフトウェアの意味論も、要件からのソフトウェア自動生成プロセスも、形式的でない言語で説明されているので、この方法論を理解し研究しようとすると、難しく、混乱が生じる恐れがある。さらに、この方法論の背後にあるアイデアは、このようなアイデアをサポートするのに適当でない、逐次処理言語(sequential language)によって記述されている。実際、Lyeeにより生成したソフトウェアは、基本的には、小さなコンポーネント(Lyee用語でベクトル(vector)と呼ばれる)群からできている。ここでは、各コンポーネントはそれぞれのアトミック・ゴール(atomic goal:要素の目的)を持って相互作用によって互いに協力し合い、要求される結果(グローバル・ゴール(global goal:全体の目的))を生成する。一方、平行性(concurrency:複数の計算主体が互いに情報のやりとりを行いながら動作すること)および通信(communication)をサポートするプロセス代数は、共同作用しあう(concurrent)コンポーネントを記述するのに本質的に適した言語であることはよく知られている。それ故、プロセス代数は、Lyee方法論の概念をサポートするための、形式的で価値ある基礎を提供する。Lyee方法論にプロセス代数を使用することによる数ある利点のなかには、Lyeeの現実のバージョンにおける多くのコンポーネント(ベクトル全体またはベクトルの一部)は、もはや必要でなくなる、ということがある。これらのコンポーネントは、逐次処理言語によって記述されているがために、ある処理のまとまりの実行順序を制御するためにどうしても必要であったが、これらの役割は、プロセスの平行処理と通信をサポートするプロセス代数により、必然的にサポートされるからである。実際、経路作用要素(routing vector)は、もはや必要ではなくなる。より小さく、処理時間がより短く、必要メモリがより少ないプログラムを生成する、シンプルなLyeeの形式的記述を手にすることができる。その上、この形式的記述は、この方法論のいろいろな局面の多くの興味ある分析に不可欠なスタート点になる。例えば、Lyeeで

生成したソフトウェア、またはソフトウェア生成プロセスを最適化するために、最適化されたプログラムが元のバージョンと同じものであるという形式的証拠を必要とする。プロセス代数を使用すれば、形式的に同じものであることのチェック、またはもっと一般的にモデル・チェックをうまく行うことができる。

[0017] 本稿においては、最初に、Lyee方法論の基本的なコンセプトを容易かつ本質的(naturally)にサポートする、Lyee計算法(Lyee-Calculus)と呼ぶ形式的プロセス代数(formal process algebra)を定義する。実際に、この計算法は、フォン・ノイマン型(逐次処理型)の計算機より、Lyee方法論の概念をサポートするのにより適した抽象機械(abstract machine:計算機の概念を抽象化したもの)と見なすことができる。この機械は、プログラムを、最終結果を生成するために共に相互作用を行う分子の集合と見なす。第二に、この計算法が、どのようにして、Lyeeコンポーネント(Lyeeではベクトルと呼ぶ)とLyee方法論の全ソフトウェア生成プロセスの両方を形式化し、簡単にすることができるのかを示す。実際に、Lyeeソフトウェア生成の自動生成全プロセスを形式化した。

[0018] 本稿の残りの部分は、下記のような構成になっている。セクション2において、Lyee計算法の構文論(syntax)および意味論(semantics)を定義する。セクション3においては、Lyee方法論の技術的紹介を行い、Lyee計算法によるその形式化の全体像を紹介する。セクション4においては、Lyee計算法によるLyee方法論の詳細かつ完全な形式化を提案する。セクション5においては、生成したプログラムの、実行段階毎の記述と一緒に、どのようにして簡単な要件からプログラムが自動的に生成されるのか、の両方を具体的に示す1つのケーススタディについて説明する。最後に、セクション6においては、この研究の結論を述べ、将来の研究のいくつかについて説明する。

2. Lyee計算法

プロセス代数は、複雑なコンピュータ・システム、特に、通信し、平行に実行するコンポーネントを含むコンピュータ・システムのための形式的記述技術である(非特許文献1、3、5参照)。プロセス代数は、通常、文献で計算法(calculus)と呼ばれる。何故なら、プロセス代数は、プロセス代数の分野を超えて、種々の数学的および論理的概

念(平行性理論(concurrency theory)、動作意味論(operational semantics)、複雑性理論(complexity theory)、ロジック等)を含んでいるからである。

[0019] このセクションにおいては、Lyee方法論を形式化するために特に定義したLyee計算法と呼ぶ新しい計算法の構文法および意味論について説明する。

2.1 構文法(Syntax)

Lyee計算法プログラムは、指定されたチャネル上でハンドシェイク技術(hand-shake technique:2つのプロセスの一方を受信可能状態に、他方を送信可能状態におくことによって通信させる技術)により通信(communicate)する独立した並行プロセスからなるシステムである。チャネルは、その上では特定のプロセスだけが通信することができるように制限できる。あるプロセスは、作用[?!e]を行うことにより、チャネル?を通して値?を送ることができる(?は、式eの計算結果(valuation))。同様に、あるプロセスは、作用[??e]を行うことにより、チャネル?から式eにの値?を受信することができる(?は、式eの計算に使用する値)。

[0020] あるプロセスは、また、不作用作用(silent action) τ を行うこともできる。この特別な作用は、プロセス間の同期(synchronization:プロセス間通信において、あるプロセスが値などを送信したと同時に、他のプロセスがそれを受信すること)のような、システム内部のふるまい(internal behavior)を表わすことを意図している。また、プロセスの進展における予見不能性(indeterminism)、すなわち、複数のプロセスの選択肢があってどのプロセスが実行させれるか予測できないような場合(後述のプロセス構文法定義1の「選択」の説明を参照)、を捉えるために有益である。

[0021] プロセスの構文法(syntax)を以下に定義する。

＜定義＜構文法1＞

[0022] [数1]

$$P,\ Q ::= [K].P \mid (P \mid Q) \mid P + Q \mid P \triangleright Q \mid P/L \mid A(\vec{X}) \stackrel{def}{=} P \mid nil$$

＜定義＜構文法2＞

$K, K_1, K_2 ::= \{\kappa\} \mid K_1 \cup K_2$

＜定義＜構文法3＞

[0023] [数2]

$$L, L_1, L_2 ::= \quad 0 \mid \{\iota\} \mid L_1 \cup L_2 \mid$$

＜定義＜構文法4＞

κ ::=??!e | ??e | τ |

上記の定義構文法に使われる記号の意味と、その直感的構文法の意味は下記のとおりである。

＜定義＜構文法1＞

[0024] [表1]

記号	意味
P, Q	プロセス (processes)
::=	左辺を右辺のように定義する
|	または
[K].P	シーケンス (Sequence)
K	作用の集合
P | Q	並行構成 (Parallel composition)
P + Q	選択 (Choice)
P▷Q	条件付き選択 (Guarded choice)
P/ L	制限 (Restriction)
$A(\vec{X}) \stackrel{def}{=} P$	定義 (Definition)
nil	無プロセス (nil process)

定義構文法1の直感的意味は次のとおりである。すなわち、プロセスPとQがあったとき、その関係や内容は、以下のいずれかとして定義される。すなわち、シーケンス([K].P)、並行構成(P | Q)、選択(P + Q)、条件付き選択(P>

Q)、制限(P/ L)、定義、無プロセス(nil)、のいずれかである。ここで、「定義」とは、

[0025] [数3]

$$\stackrel{def}{=}$$

のことで、記号の左辺は、右辺によって定義される。

[0026] 各構文の解釈については後述する。

＜定義＜構文法2＞

[0027] [表2]

記号	意味
K, K_1, K_2	作用の集合（Set of actions）
κ	作用（Action）
$\{\kappa\}$	1つの作用（Single action）を要素とする集合
$K_1 \cup K_2$	作用の集合の結合集合（Set union）

定義構文法2の直感的意味は次のとおりである。すなわち、作用の集合K、K_1、K_2があるとき、その作用の集合は、1つの作用を要素とする集合（$\{\kappa\}$）、作用集合が結合した集合（$K_1 \cup K_2$）、のいずれかである。

＜定義＜構文法3＞

[0028] [表3]

記号	意味
L, L_1, L_2	チャネルの集合（Set of channels）
0	空集合（Empty set）
ι	チャネル（Channel）
$\{\iota\}$	1つのチャネル（Single channel）を要素とする集合
$L_1 \cup L_2$	チャネルの集合の結合（Set union）

定義構文法3の直感的意味は、次のとおりである。すなわち、チャネルの集合L, L_1, L_2があるとき、そのチャネルの集合は、要素を持たない空集合（0）、1つのチャネルだけを要素とする集合（{?}）、チャネルの集合が結合した集合（$L_1 \cup L_2$）、のいずれかである。

＜定義＜構文法4＞

[0029] [表4]

記号	意味
κ	作用（Action）
ι!e	送信（Send）作用 （チャネルιを通して、数式eの計算結果の値を送信する）
ι?e	受信（Receive） （チャネルιを通して、数式eの値を受信する）
τ	不作用作用（Silent action）
$\iota, \jmath$	チャネル（Channel）
e	数式（Arithmetic expression）

定義構文法4の直感的意味は、次のとおりである。すなわち、作用κがあるとき、その作用は、チャネル?上で式eの計算結果の値を送信する送信作用（?!e）、チャネル?上で式eの値を受信する受信作用（??e）、不作用作用（τ）、のいずれかである。

[0030] 定義構文法1に定義した、構文法の意味を詳しく説明する。

1)[K].P:シーケンス(Sequence)

プロセス[K].Pは、作用の集合であるKに属す全ての作用(action)、すなわち{κ_1, . . . , κ_n}、を実行した後にPとしてふるまうプロセスである。Kに属する作用を実行する順序は重要ではない。ここでは、説明を簡単にするために、[{κ_1, . . . , κ_n}]と書く代わりに[κ_1, . . . , κ_n]と書く。

2)P | Q:並行構成(Parallel composition)

プロセスP | Qは、並行に実行されるプロセスPおよびQとしてふるまう。各プロセスは、両方が知っているチャネル上で互いに相互作用することができるし、または各プロセスが他方から独立して外部世界(システムの環境またはエンドユーザ)と相互作用(interact)することもできる。同じチャネル上で2つのプロセスが同期する(synchronize:プロセス間通信において、あるプロセスが値などを送信したと同時に、他のプロセスがそれを受信する)と、プロセス全体は、作用 τ (不作用作用)を実行し、その後、残りのプロセスとしてふるまう。具体的に、下記のプロセスの例で考えてみる。

[0031] P=P1 | P2

P1およびP2を下記のように定義する。

[0032] [数4]

$$P1 \stackrel{def}{=} [\imath!4].P1'$$
$$P2 \stackrel{def}{=} [\imath?x].P2'$$

上記式の直感的意味は、

P1は、値4をチャネル?上に送信し、その後は、プロセスP1'としてふるまうプロセスである。P2は、チャネル?上に式xの値が与えられるのを待って、値が与えられたら受信し、その後は、プロセスP2'としてふるまうプロセスである。

従って、プロセスPは、プロセスP1とP2が同期したとき(それは実際にはP1からの値4の送信と同時にP2による受信が行われることによる)、不作用作用 τ を実行し、その後は、残りのプロセスP1' | P2'としてふるまう。意味論上、この遷移を下記のように表わす。

[0033] [数5]

$$P \xrightarrow{\tau} P1' \mid P2'$$

この式の直感的意味は、プロセスPにおける相互作用の結果、不作用作用 τ が実行され、プロセス全体が、並行プロセスP1' | P2'に移行する、ということである。

[0034] 関係

[0035] [数6]

$$\xrightarrow{\kappa}$$

の形式的定義はこのセクションの後の方で示す。

3)P+Q:選択(Choice)

プロセスP+Qは、プロセスPまたはQとしてふるまうプロセスである。PおよびQのどちらもが、始まりが不作用作用であるような場合を除いて、どちらのプロセスが実行されるかの選択は、環境によって決定論的に決まる。一方、両方のプロセスが不作用作用で開始する場合には、選択は決定されない。

4)P>

Q:条件付き選択(Guarded choice):

プロセスP>

Qは、プロセスQが活性化(activated)されるまで、Pとしてふるまうプロセスである。後者Qが活性化されたときはいつでも、Pは停止し、メモリからクリアされる。

5)P/L:制限(Restriction)

プロセスP/Lは、チャネルの集合Lに与えられたチャネルだけを使って環境と通信することができるプロセスP、としてふるまうプロセスである。

6)数3の記号:定義(Definition)

たとえば、数3の記号によって、下式のような定義ができる。

[0036] [数7]

$$A(\vec{X}) \stackrel{def}{=} P$$

数7の式において、Aはプロセスの識別名、

[0037] [数8]

$$\vec{X}$$

はプロセスAの変数(パラメータ)で、左辺のプロセスは、右辺のプロセスPに等しいと定義されている。プロセスPは再起的にAを含むことができる。

7)nil:無プロセス(nil process):

どんな作用も実行できないプロセス、いいかえると終了しているプロセス(dead process)である。あるプロセスが終了すると、通常nilプロセスになる。終了してnilプロセスになったとき、そのプロセスはもはや起動して(activated)いないので、作用を実行する条件が整ったとしても、何の作用も実行できない。

2.2 例値を保持するプロセスのモデル化

このセクションにおいては、メモリという、値を保持するセル(cell)を、その通信チャネルを通して、その環境と相互作用を行うプロセスとしてモデル化する方法について説明する。図1に示すように、メモリ・セルは通信のための2つのポート(チャネル)inおよびoutを持つものと見なす。このメモリ・セルの基本的なタスクは、チャネルin上で値を無制限に待つことであり、チャネルout上でその値を利用可能にすることである。メモリ・セルは、新しい値がチャネルinに入力されるまで、同じ値を必要な回数だけチャネルoutに出力することができる。チャネル自体は値を保持しない。

[0038] メモリ・セルxが値?を保持しているプロセス・セル$C^x(?)$は、下記のように表わす。

[0039] [数9]

$$C^x(v) \stackrel{def}{=} [in^x?y].C^x(y) + [out^x!v].C^x(v)$$

この形式的記述の直感的意味は、値?を保持しているプロセス$C^x(?)$は、チャネルin^x上で式yの値を受信したときは、プロセス$C^x(y)$としてふるまい(すなわち保持する値が?から式yの値に変更される)、保持している値?をチャネルout^x上に送信したときには、送信作用の後も値?を保持したままプロセス$C^x(?)$としてふるまうプロセスである。

[0040] 値を保持するメモリ・セルをプロセスとして捉えることにより、メモリ・セルにさらに知能を付加することができる。例えば、初期化される(初期値を受信する)までその内容にアクセスさせないセルを書くことができる。情報処理能力のあるこのスマート・セル(smart cell)は、下記のように定義することができる。

[0041] [数10]

$$Cell(x) \stackrel{def}{=} [in^x?y].C^x(y)$$

プロセスCell(x)は、チャネルin^x上で式yの値を受信した後、式yの値を保持するプロセス$C^x(y)$としてふるまうプロセスである(式yに値を受信し、式yの値を保持するプロセスで、送信は行わない)。

[0042] ここで、下記のように定義した2つのプロセスについて考えてみる。

[0043] [数11]

$$P1 \stackrel{def}{=} [in^x!5].nil$$
$$P2 \stackrel{def}{=} [out^x?y].nil$$

プロセスP1は、チャネルin^x上に値5を送信し、nilになる(終了する)プロセスである(値5を送信するプロセス)。プロセスP2は、チャネルout^x上で式yの値を受信した後にnilになる(終了する)プロセスである(数式yに値を受信するプロセス)

これらの2つのプロセスが、セルx(C^x)を通して通信するようなプログラムを書くことが容易にできる。

[0044] P1 | Cell(x) | P2

このプログラムは、プロセスP1、プロセスCell(x)、プロセスP2が並行に実行されるプログラムである。

[0045] 図2は、上記に含まれる全てのプロセス間の相互作用を示したものである。

[0046] このプログラムの実行における、個々のステップの詳細は以下のようになる。

[0047] [数12]

$$\begin{aligned} P1 \mid Cell(x) \mid P2 &= \underline{[in^x!5].nil} \mid \underline{[in^x?y]}.C^x(y) \mid [out^x?y].nil \\ &\xrightarrow{\tau} C^x(5) \mid [out^x?y].nil \\ &= ([in^x?y].C^x(y) + \underline{[out^x!5]}.C^x(5)) \mid \underline{[out^x?y]}.nil \\ &\xrightarrow{\tau} C^x(5) \end{aligned}$$

上記の式の意味を直感的に表わせば、プロセスP1 | Cell(x) | P2は、以下の第1ステップと第二ステップを経て実行される。(等号記号で記載される右辺が1つのステップである)下線は相互作用(同期)を起こす送受信作用を示している。

＜第1ステップ＞

(1)[in^x!5].nil(チャネルin^xに値5送信)、(2)[in^x!5].C^x(y)(チャネルin^xで式yに与える値を受信し、受信後はセルxに式yの値を保持)、(3)[out^x?y].nil(チャネルout^xで式yの値を受信)、の3つのプロセスが並行に行われた結果、

(1)がチャネルin^xに値5を送信完了し、同時に(2)がチャネルin^xで値5を受信する、というプロセス間の相互作用(同期)が起こって、

不作用作用 τ が実行され、

(4) C^x(5)(セルxに値5を保持)、と(3)[out^x?y].nil(チャネルout^xで式yの値を受信)、の2つのプロセスの並行プロセスに移行する。

＜第2ステップ＞

前述の数1－19の定義から、(4) C^x(5)は選択プロセス(4)－1 ([in^x?y].C^x(y)＋[out^x??].C^x(?))に置き換えることができる。

すなわち、

プロセス(4)－1と、(3)が並行に行われた結果、

(4)－1の[out^x??].C^x(?)(チャネルout^xに値5を送信)と同時に(3)の(チャネルout^xで式yの値5を受信)が起こる、という相互作用(同期)が起こって、

不作用作用 τ が実行され、

その結果、残りのプロセスはC^x(5)(セルxに値5を保持)としてふるまわれる。

2．3 意味論

以下に、Lyee計算法の形式的意味論について説明する。この意味論は、相互作用関係

[0048] [数13]

$$\overset{\kappa}{\rightarrow}$$

により定義される。この時、演算子→は「関係」(relation)を表わし、κ は作用を表わす。「関係」は複数の「関係」の集合であり、その集合要素である1つ1つの「関係」は次の3つの要素で構成されている。第1の要素は「変化前のプロセス」、第2の要素は「変化の後のプロセス」、第3の要素は「変化の間に実行された作用」である。

[0049] [数14]

$$P \overset{\kappa}{\rightarrow} Q$$

と書くとき、関係→の3要素は(P, Q, K)である。

[0050] 上記のように定義するとき、数14の意味は、サブプロセスPの中の相互作用(reaction)によって、全体プロセスがアトミック・アクション(atomic action;割込みを許さない一連の不可分な作用で、次の別の作用を開始させるために、必ず完了されなければならない)κを実行してサブプロセスQとなる、ことを意味する。相互作用関係(interaction relation)という着想は、ベリー(Berry)およびバウドル(Boudol)(非特許文献2参照)の化学抽象機械(Chemical Abstract Machine)によって触発されたものである。このモデルでは、プロセスは、相互作用を起こすことを待っている分子の化学溶液であると見なしている。

[0051] 数式13の関係を形式的に表わすために、下記の概念を定義する必要がある。

[0052] [数15]

$\leadsto$

は、プロセスを簡単にするために用いる記号で、左辺が、簡素化されたプロセスである右辺と等しいことを表わす。下記に示すように、無プロセスを除去することによりプロセスを簡単にすることができる。

[0053] [数16]

$$P \mid nil \leadsto P \qquad nil + P \leadsto P \qquad nil \rhd P \leadsto P$$
$$nil \mid P \leadsto P \qquad P + nil \leadsto P \qquad P \rhd nil \leadsto P$$
$$nil/L \leadsto nil$$

式eの計算結果である値の集合を

[0054] [数17]

$[\![e]\!]$

で示す。eが変数である場合には、その計算結果の値の変域も、その変数の変域(domain)(整数、実数等)と同じである。説明を簡単にするために、すべての変数は実数の集合に属するものと見なす。

[0055] [数18]

$\kappa_!$

は、作用 κ が使用するチャネルの名前である。たとえば、

[0056] [数19]

$$\begin{aligned} \tau_! &= \emptyset \\ (a!e)_! &= (a?e)_! = \{a\} \end{aligned}$$

の意味は次の通り:

不作用作用 τ が使用するチャネルは存在しないので、空集合(0)である。

送信作用(?!e)と受信作用(??e)が使用するチャネルは、?である。

[0057] 「関係」の集合(→)は、要素として、表5に示す15個の規則を満足させるような「関係」を最小数持つ「関係」である。

[0058] かっこ内ないに示した各種のR記号は、規則名を示す記号である。規則は、前提条件と結果からなっており、各規則は、横線の上段に示される条件のとき、下段に示した結果になるような規則である。

[0059] [表5]

$$(R_{\rightarrowtail}) \frac{P \xrightarrow{\kappa} Q' \quad Q' \rightarrowtail Q}{P \xrightarrow{\kappa} Q}$$

$$(R_!) \frac{v \in [\![e]\!]}{[a!e].P \xrightarrow{a!v} P} \qquad (R_?) \frac{v \in [\![e]\!]}{[a?e].P \xrightarrow{a?v} P[v/e]} \qquad (R_\tau) \frac{v \in [\![e]\!]}{[\tau].P \xrightarrow{\tau} P}$$

$$(R_{[\,]}) \frac{[K_1].P \xrightarrow{\kappa} P'}{[K_1 \cup K_2].P \xrightarrow{\kappa} [K_2]P'} \; K_2 \neq \emptyset$$

$$(R^l_+) \frac{P \xrightarrow{\kappa} P'}{P + Q \xrightarrow{\kappa} P'} \qquad (R^r_+) \frac{Q \xrightarrow{\kappa} Q'}{P + Q \xrightarrow{\kappa} Q'}$$

$$(R^\tau_\triangleright) \frac{P \xrightarrow{a?v} P' \quad Q \xrightarrow{a!v} Q'}{P \triangleright Q \xrightarrow{\tau} Q'} \qquad (R^l_\triangleright) \frac{P \xrightarrow{\kappa} P'}{P \triangleright Q \xrightarrow{\kappa} P' \triangleright Q} \qquad (R^r_\triangleright) \frac{Q \xrightarrow{\kappa} Q'}{P \triangleright Q \xrightarrow{\kappa} Q'}$$

$$(R^\tau_|) \frac{P \xrightarrow{a?v} P' \quad Q \xrightarrow{a!v} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad (R^l_|) \frac{P \xrightarrow{\kappa} P'}{P \mid Q \xrightarrow{\kappa} P' \mid Q} \qquad (R^r_|) \frac{Q \xrightarrow{\kappa} Q'}{P \mid Q \xrightarrow{\kappa} P \mid Q'}$$

$$(R_=) \frac{P[\vec{Y}/\vec{X}] \xrightarrow{\kappa} P'}{A(\vec{Y}) \xrightarrow{\kappa} P'} \; A(\vec{X}) = P \qquad (R_/) \frac{P \xrightarrow{\kappa} P'}{P/L \xrightarrow{\kappa} P/L} \; \kappa_! \in L$$

たとえば、第2番目の規則($R_!$)の意味は以下のとおりである:

値?が式eの値の集合に属すとき(上段の前提条件が成立するとき)、チャネル?上に式eの値を送信後、プロセスPとなる[?!e].Pは、チャネル?上への値?送信作用が起こ

った後、プロセスPに移行する(下段の結果)。

3. Lyee要件の非形式的形式化

このセクションにおいては、Lyee方法論が、どのようにして基本的なユーザ要件からソフトウェアを生成するのか、その全体像について説明する。加えて、この全体像から、Lyee計算法によってこの方法論をどのようにして簡単かつ効率的に形式化することができるのかを順次紹介する。Lyee方法論のソフトウェア生成プロセスの完全な形式化については、次のセクションで説明する。

3. 1 Lyee要件

Lyee方法論において、要件は、宣言的な方法で、単語名、その定義式、その計算条件、およびその属性(入出力、タイプ、セキュリティタイプ等)という要素を含む宣言の集合として与えられる。表6は、Lyee要件の一例である。

[0060] 「Word」は単語名、「Definition」は単語の値を生成するための定義式、Conditionは定義式を実行する計算条件、「IO」は単語の値の入出力の属性を示す。OFはファイルへの出力、OSは画面への出力、ISは画面からの入力、IFはファイルからの入力(表にはない)、を表わす。「Type」は、値の属性で、intは整数(integer)、floatは不動小数点数を表わす。「Security」は、値のセキュリティに関する属性を表わす。secretは非公開、publicは公開、を表わす。

[0061] [表6]

単語名	定義式	計算条件	IO	タイプ	セキュリティ
		⋮			
a	b+c	b*e>2	OF	int	secret
c			IS	float	public
b	2*c+5	c>0	OS	float	public
e			IS	float	public
		⋮			

表6の要件は、感覚的にとらえれば、従来のプログラミング言語の表7のコードに対応する。S_aは、単語aの宣言を意味する。

[0062] [表7]

宣言	コード

s_a	if b*e>2
	then a:=b+c; output(a);
	endif

s_c	input(c);

s_b	if c> 0
	then b:=2*c+5; output(b);
	endif

s_e	input(e);

たとえば、単語aの宣言S_aのコードは、

「もし、b*e>2が真ならば、単語aにb+cの計算結果を代入し、単語aの値を出力」、を意味している。

[0063] 宣言S_cのコードは、「単語cに値を入力」、を意味している。

[0064] Lyee方法論においては、ユーザは、これらの定義が実行される順序(制御ロジック)を指定する必要はない。表7に示すように、単語aの定義は、単語bを使用しているにもかかわらず、宣言S_bは、宣言S_aの後に位置している。以下に説明するように、これらの要件から、また、宣言の順序とは無関係に、Lyeeはすべての定義した単語を計算するコードを生成することができる。このシンプルなアイデアは、非特許文献6、7、8、9に示されるように、ソフトウェア開発上の異なるステップにおいて、数々の有利な結果を生む。実際、このアイデアを使用することにより、要件が不完全であってもソフトウェアの開発をスタートすることができる。さらに、ユーザは、もっと古典的な方法論の場合のように、単語の実行順序の制御ロジックの問題を扱う必要がない。ソフトウェアの制御ロジック部分は、Lyee方法論においては、自動的に生成され、その結果、プログラミング・エラーが低減し、プログラミング時間が短縮される。柔軟性も、Lyee方法論の主要な利点である。何故なら、保守業務を、要件の簡単な修正(単語の定義の追加、削除および/または修正)作業にまで軽減できるからである。

[0065] 従って、Lyeeシステムは、求められる結果を生成するために共に相互作用を行う独立したコンポーネント(宣言)の集合体と見なすことができる。本論のプロセス代数の概念は、この見方と非常によく一致する。何故なら、Lyee計算法は、プログラムを、最終目的に到達するために、共に相互作用しあう分子(プロセス)の化学溶液と見なす

からである。それ故、一見して、宣言の集合{s_1, . . . , s_n}からなるLyee要件LRを、Lyee計算法の並行プロセス(concurrent process)として見ることができる。すなわち、次に表わすとおりである。

[0066] $$LR = s_1 \mid \ldots \mid s_n$$

直感的説明によれば、要件LRは、プロセスs_1、. . . 、s_nが並行に実行されるプロセスである。

3. 2 パレット(Pallet)および基本構造(Scenario Function)

表6の要件から、aおよびbの値を計算し、それらを出力するプログラムを、自動的に生成することができる。このプログラムは、不動点(fixed point)に達するまで、すなわち、図3に示されるように、繰り返し処理がいかなる単語の値も変えなくなるまで、これらの命令の実行を単純に反復する。

[0067] プロセス代数の観点からいうと、「不動点に達する」という概念は、当然意味規則に組み込まれる。実際に、プロセス(分子)は、いかなる進展も可能でなくなる状態(不動点)に達するまで、共に相互作用する。

[0068] 要件からLyeeにより自動的に生成されるプログラムの構造および内容についてさらに正確に説明する。Lyee方法論においては、表6に示すような宣言の集合の実行は特定の方法により行われる。実際は、Lyeeは、図4に示すように、Lyee用語で、パレット(W_{02}, W_{03}およびW_{04})と呼ばれる3つの領域上に宣言に関連するコードを分配する。

パレットW_{02}:
入力単語を処理する。

パレットW_{03}:
単語の計算条件を計算し、結果をブール型(真偽値を値としてもつ)の変数で保存する。例えば、単語aの定義の中で使用されている計算条件「b*e>2」はW_{03}で計算され、真/偽の結果は、別の変数「a__cond」に保存される。

パレットW_{04}:
要件に含まれるその定義式によって単語の計算を行う。パレットW_{04}は、また、計算し

た単語の値を出力する。

[0069] Lyeeプログラムは、パレットW_{04}からスタートして、不動点に達するまで、すべての定義した単語の値を計算しようとする。単語の値の計算に係わるW_{04}内に進展が見られなくなると、経路作用要素R4によりパレットW_{02}に制御が与えられる。今度は、この第2のパレットが、単語の値の入力を繰り返し、不動点に達した(他に新たな入力対象がなくなった)とき、次に、経路作用要素R2が制御をパレットW_{03}に渡す。最後に、そしてパレットW_{04}と同じように、パレットW_{03}が、不動点に達するまで、要件に従って単語の計算条件を計算しようとする。図5に示すように、この全プロセス($W_{04}\rightarrow W_{02}\rightarrow W_{03}$)は、全体が安定状態に達するまで繰り返され、連結されている3つのパレットは基本構造(Scenario Function)と呼ばれる。

[0070] Lyee要件を組み込むために逐次処理言語を使用することによって、Lyee方法論の作者は、パレットが実行される順序、同じパレット内のベクトル(モジュール)が実行されるべき順序、およびあるベクトルから他のベクトルに制御を渡す方法を明確に指定せざるをえなかった。言い換えれば、この逐次処理言語がゆえに、Lyee方法論の基本概念に属さない、また、この方法論をかなり複雑にしてきたいくつかの局面がもたらされることになった。すなわち、各パレット内のベクトルが実行される繰り返しの制御、パレットから他のパレットに実行を移す(異なる基本構造間を含め)方法および時期等を指定するという必要があった。これらのために、パレット関数(パレット内の実行制御)、経路作用要素(次に実行するパレットの決定)、パレット連鎖関数(パレット間の実行制御)が設置されていた。

[0071] しかし、Lyee計算法を使用することにより、パレットが実行される順序、各パレットのベクトルが実行される順序、パレットから他のパレットに制御を渡す方法および時期等を指定する必要がもはやなくなる。このような詳細は、当然ながら本論の抽象機械Lyee計算法により本質的に管理される。それ故、基本構造は、下式のように形式化することができる。

[0072] $$SF=W_{04}\mid W_{03}\mid W_{02}$$

すなわち、基本構造SFのプロセスは、W_{04}、W_{03}、W_{02}の3つのパレットのプロセスの並行プロセスである。

[0073] W_{04}、W_{03}およびW_{02}を実行する順序はもはや重要ではなくなり、これらパレットを接続している経路作用要素も除去されていることに留意されたい。

[0074] Lyeeは、固定構造を持つ簡潔なプログラム(Lyee用語で述語ベクトル(predicate vector)と呼ばれる)を確立した。この構造により、生成されたコードの構造は不変かつ要件の内容に依存しないものになっている。ベクトルの実行を制御する統括プログラム(global program)であるパレット関数の役割は、単に述語ベクトルを呼び出すだけである。図6は、述語ベクトルの構造を示す。

[0075] 述語ベクトルの目的は、各パレットにより異なる。

<パレットW_{04}>

パレットW_{04}の第1の目的は、定義式により単語に値を与えることである。この役割を行う述語ベクトルをL4と呼ぶ。図4の例における単語aおよび単語bのL4は、図7に示すようになる。

[0076] 単語の計算に進展がなくなり、かつ、単語の値が全て成立すると、Lyeeの生成コードは、次の目的となる単語の出力を実行しようとする。値を出力するという目的を持つ述語ベクトルは、出力作用要素(Output Vector:O4と呼ぶ)と呼ばれる。他に、領域のクリアを行う構造作用要素(S4と呼ぶ)がある。

<パレットW_{02}>

値を入力単語に関連付けるという目的を持つ2つの述語ベクトルがある。メインメモリ上(Lyeeプログラム領域の外)への入力を行う入力作用要素(Input Vector:I2と呼ぶ)と、属性チェックとLyeeプログラム領域内への入力を行うL2である。

<パレットW_{03}>

パレットW_{03}の述語ベクトルL3の目的は、図8に示すように、要件内に指定されている、単語の定義式(すなわちL4)を実行するための条件を判定することである。

[0077] 最後に、表8は、表6の要件を実装したLyeeプログラムを示す。

[0078] 各パレットに1つおかれたパレット関数が、パレット内の述語ベクトルをコール(起動)する。パレット関数は、プログラムに1つおかれたパレット連鎖関数によってコールされる。パレット連鎖関数は、各パレットのR4、R2、R3の指定に従って、該当するパレット関数をコールする。その他のプログラムの意味は、表8のコメントのとおりである。

述語ベクトルL4、L2、L3(総称して論理要素と呼ぶ)の処理対象は単語単位であり、入出力作用要素(I2およびO4)、構造作用要素(S4)の処理対象は単語の集合である。

[0079] [表8]

パレット	プログラム	コメント
W_{04}	Call S4 Do Call L4 a Call L4 b while a fixed point is not reached Call O4 Call R4	領域のクリア 単語 a の定義式を計算 単語 b の定義式を計算 不動点に達するまで繰り返し実行 結果を出力 次経路としてW_{02}を指定
W_{02}	Call I2 Do Call L2 e Call L2 c while a fixed point is not reached Call R2	入力(メモリ上へ) 単語 e を入力 単語 c を入力 不動点に達するまで繰り返し実行 次経路としてW_{03}を指定
W_{03}	Do Call L3 a Call L3 b while a fixed point is not reached Call R3	 単語 a の実行条件 a_cond を計算 単語 b の実行条件 b_cond を計算 不動点に達するまで繰り返し実行 次経路としてW_{04}を指定

ここでLyee計算法を使用すると、W_{04}、W_{03}およびW_{02}のプロセスは、それぞれ下記のように定義することができる。

[0080] [数20]

$$W_{04} = S4 \mid L4\_a \mid L4\_b \mid O4$$

$$W_{03} = L3\_a \mid L3\_b$$

$$W_{02} = L2\_e \mid L2\_c \mid I2$$

W_{04}プロセスは、S4、L4_a、L4_b、O4の各プロセスの並行プロセスである。

W_{03}プロセスは、L3_a、L3_bの各プロセスの並行プロセスである。

W_{02}プロセスは、L2_e、L2_c、I2の各プロセスの並行プロセスである。

異なる述語ベクトル(L2、L3、L4、I2、O4、S4)の形式的定義については、次のセクションで説明する。

3.3 処理経路図(Process Route Diagram)

前のセクションで示した基本構造は、任意の、要件が簡単なケースのプログラム全体だと見なしてもよい。特に、すべての入力単語および出力単語が、同じ画面に属していて、データベースが全く使用されていない場合のプログラムである。入力単語および出力単語が、いくつかのデータベースまたは、相互に接続された異なる画面に属する場合には、状況はもっと複雑になる。説明を簡単にするために、以下の説明においては、画面の数が複数あるだけの場合について説明する。図9に示すように、相互に接続している画面が3つあり、ユーザは画面から別の画面に移動することができる場合について考える。各画面において、ユーザは単語の入力、計算、また出力ができる。従って、仕様書では、ユーザは、これらの画面がどのように相互接続しているのかを指定しなければならない。

[0081] さらに言えば、1つの基本構造だけを定義して、そこで全画面中の定義された単語全てを計算するのは都合がよくない。なぜなら、実際、プログラムの任意の実行において、ある画面は訪問されないかもしれず、その場合、訪問されなかった画面の単語の値の計算が無駄になるからである。そのため、Lyeeは、各画面に、その画面が訪問された場合だけ実行されるような担当基本構造を関連づける。画面に係わる基本構造は、1つの画面から他の画面への移動を示すように、接続されている。Lyee用語において、複数の基本構造が接続されたものが、図10に示すような、処理経路図(Process Route Diagram)である。

[0082] Lyee計算法を使用して、ある画面s_kを、プロセスSF(s_k)の実行を制御するプロセス$\Phi(s_k)$、として下式のように形式化できる。

[0083] [数21]

$$\Phi(s_k) = SF(s_k) \triangleright [i^{s_k}?1].\Phi(s_k)$$

すなわち、プロセス$\Phi(s_k)$は、プロセスSF(s_k)(画面s_kに係わる処理の基本構造のプロセス)としてふるまい、チャネル$?^{sk}$上で真の値を受信したとき、SF(s_k)を停止してプロセス$\Phi(s_k)$としてふるまう。実際の動きとしては、制御プロセス$\Phi(s_k)$は、プロセスSF(s_k)を活性化し、画面s_kに関連するチャネル$?^{sk}$上で信号を受信したとき(すなわち画面s_kが再活性化されたとき)、$\Phi(s_k)$自身が再活性化される。従って、制御プロセス$\Phi(s_k)$が再活性化されることによって、SF(s_k)は一度終了して再活性化され、初期状態(fresh

instance)に戻る。

[0084] 関数$\Phi(s_k)$がどのように機能するかを示す具体例は、本明細書の後の方に記す。

3．4 Lyeeプログラム

まとめると、Lyee方法論によるLyeeプログラムの構造は次のとおりである。Lyeeプログラムはいくつかの処理経路図(PRD)で構成される。各PRDは、相互に接続された基本構造(SF)の集合である。各基本構造は、3つの相互に接続されたパレットW_0、W_{02}、W_{03}、W_{04}で構成される。最後に、パレットは、述語ベクトルによって構成される。述語ベクトルには、Lyeeプログラムの最小単位のモジュールであるので、アトミック・ベクトル(atomic vector)とも呼ぶ。述語ベクトルには、論理要素(signification vector:L2、L3、L4の総称)および作用要素(action vector:I2、O4、S4、R2、R3、R4の総称)がある。これらの述語ベクトルの実行を制御する制御モジュールとして、パレットごとにパレット関数がおかれ、そのパレット関数の実行を制御するモジュールとして、プログラムに1つのパレット連鎖関数がおかれる。

[0085] Lyee計算法を使用することにより、画面$s_1, \ldots, s_k$を含むLyeeプログラムPは、下式のように形式化される。

[0086] [数22]

$$\mathcal{P}(s_1, \ldots, s_k) = \Psi(s_1, \ldots, s_k)/\mathcal{L}(s_1, \ldots, s_k)$$

ここで、$L(s_1, \ldots, s_k)$は、入力チャネルおよび出力チャネルの集合である。この式は、$\Psi(s_1, \ldots, s_k)$が、$L(s_1, \ldots, s_k)$に属すチャネルだけを通して環境とコミュニケートするように制限する。

[0087] 関数Ψは、下式のように定義される。

[0088] [数23]

$$\Psi(s_1, \ldots, s_k) = (|_{s \in \{s_1, \ldots, s_k\}} [i^s?1].\Phi(s)) \triangleright [i^{s_0}?1].nil$$

この関数Ψは、画面s(sは画面$s_1, \ldots, s_k$のいずれか)に対応するチャネル$?^s$上で真の値を受信するたびに、画面sに対応するSF(s)を起動するために、プロセスΦ(s)を起動。上記チャネル$?^s$は、対応する画面を起動するためにユーザが使用するボタ

ンやメニューを形式化したものである。また、この関数Ψは、チャネル?^{s0}上で真の値を受信した時ときに、すべての他のプロセスを終了させて、関数Ψ自身も終了(nilになる)する。チャネル?^{s0}は終了ボタン(または、対応するメニュー項目)を形式化したものである。

[0089] 従って、Lyeeシステムは、求める出力単語を計算するために相互にコミュニケートする、独立した並行プロセス(concurrent process)の集合体と見なすことができる。Lyee方法論の従来の逐次処理的(sequential)見方とは対照的に、すべての経路作用要素は必要なく、制御機能の役割は大幅に簡易化され、作業メモリ領域(working memory)等も必要ない。$\Psi(s_1, \ldots, s_k)$については、具体例と一緒に以下にさらに詳細に説明する。

[0090] 次のセクションにおいては、Lyee方法論のより詳細かつ完全な形式化について説明する。

4. Lyee方法論の形式化

Lyee計算法を使用して、簡単なユーザ要件からどのようにしてソフトウェアを自動的に生成するかを見る。

ユーザ要件は、k個の画面$\{s_1, \ldots, s_k\}$を含んでいるとする。また、各画面は宣言の集合を含んでいるとする。この場合、各宣言は、次の形式、すなわち、(w, e, c, InOut, type)を持つ。ここで、wは単語名、eはその定義式、cは定義式の実行条件、InOutは単語が入力か出力か、あるいは両方か、または入力でも出力でもないのかの指定(値iは入力単語に、値oは出力単語に、ioは入力および出力両方である場合に、空のフィールドは入力でも出力でもない場合に使用される)、typeは値の属性(タイプ)を示す(たとえばタイプBは、ボタンを示す記号として割り当てられる)。

この要件に関するプログラム$P(s_1, \ldots, s_k)$を定義するために、下記の表に示したように論理要素および作用要素のベクトル(プロセス)を形式化する。

[0091] まず、形式化に用いられている未定儀定義の記号を説明する。

Use(e)は、式eで使用する単語の集合を示すものとする。例えば、Use(a*b+1)={a, b}である。

また、F(S)は、変数Sとして入力値の集合をとり、その入力を行う受信作用の集合を

かえす関数である。

[0092] [数24]

$$\begin{aligned} \mathcal{F}(\emptyset) &= \emptyset \\ \mathcal{F}(\{x\} \cup A) &= \{i_4^x?x\} \cup \mathcal{F}(A) \end{aligned}$$

上記の意味は次のようになる。

入力値がないとき(F(0))、受信作用はない(0は空集合)。

入力値xと入力値Aの受信作用(F({x}∪A))は、チャネル

[0093] [数25]

i_4^x

上でxの値を受信する受信作用と、入力Aの受信作用F(A)の和集合である。

[0094] 論理要素(Signification Vectors)は表9に示す。

[0095] [表9]

ベクトル		コメント
L_4 (x, e) =		<定義式eを持つ単語xのL_4 (宣言の項目InOutがo) >
	$[i_3^x?1].$	チャネルi_3^x上で(L_3から)真の値(1)を受信するために待機。
	$[\mathcal{F}(Use(e))].$	メモリから、eのUseの値全てを受信するために待機。
	$[y_4^x!e].$	eの値を求め、結果をxのメモリ・セルに記憶。
	nil	終了。
L_3 (x, c) =		<定義式実行条件cを持つ単語xのL_3 (宣言の項目InOutがoまたはio) >
	$[\mathcal{F}(Use(c))].$	メモリからcのUseの値全てを受信するために待機。
	$[i_3^x!c].$	cの値(真偽値になる)を求め、結果をチャネルi_3^x上に(L_4に向けて)送信。
	nil	終了。
L_2 (x) =		<単語xのL_2 (宣言の項目InOutがi) >
	$[i_2^x?x].$	チャネルi_2^xで(入力作用要素から)xの値を受信するために待機。
	$[y_4^x!x].$	xの値をチャネルy_4^xに送信(xのメモリ・セルに保存)。
	nil	終了。

作用要素(Action Vector)は表10に示す。

[0096] [表10]

入力作用要素 (Input Vector)	コメント
I_2 (x) = $[d_x?x].$ $[v_2^x!x].$ *nil*	<{x}を要素とする入力単語グループのI_2> 入力チャネルd_x上で単語xへの値を受信するために待機。 出力チャネルv_2^x上へ(L_2に対して)単語xの値を送信。 終了。
出力作用要素 (Output Vector)	コメント
O_4 (x) = $[v_4^x?x].$ $[d_x!x].$ *nil*	<{x}を要素とする出力単語グループのO_4> メモリからxの値を受信するために待機。 出力チャネルd_xからxの値を送信。 終了。
メモリ(構造作用要素) (Structural Vector)	コメント
S_4 (x) = $[v_4^x?y].S_4^x(y)$ $S_4^x(y)$ = $[v_4^x?z].S_4^x(z)+$ $[v_4^x!y].S_4^x(y)$	<単語xのS_4> 任意の値を受信するために待機。受信するまでは、値の送信は行えない。値を受信したら(初期化と呼ぶ)、受信した値を保持して$S_4^x(y)$としてふるまう。 新しい値zを受信したときは、$S_4^x(z)$としてふるまう (セルxの内容の変更)。 保持していた値yを送信したときは$S_4^x(y)$としてふるまう。 (セルxの内容を変更しない)。 すなわち、一度値を受信して保持した後は、値の受信と保持以外に値の送信も可能となる。
経路作用要素 (Routing Vector)	コメント
R_3(b, e, s) = $[d_b?clik].$ $[\mathcal{F}(Use(e))].$ $[v^s!e].$ *nil*	<単語b(ボタンb)のR_3> ボタンbが押されるまで待機。 ボタンの条件を判定。 結果を画面sへ送信(すなわち、e＝1なら、sを起動)。 終了。

ベクトルのうち、S_4は、2.2で述べた値を保持するメモリ・セルで、かつ、「初期化される(初期値を受信する)までその内容にアクセスさせない」能力をもったスマート・セルである。

[0097] 経路作用要素は、Lyee計算法においては本質的には不要なプロセスであるが、Lyee計算法でどのように形式化できるかを示すために記載している。

パレット:

任意の画面sの3つのパレットW_{02}、W_{03}およびW_{04}は、下式のように形式化される。

[0098] [数26]

$$\begin{aligned} W_{02}(s) &= |_{(w,*,*,i,\overline{B})\in s}\, I_2(w)\ |_{(w,*,*,i,\overline{B})\in s}\, L_2(w) \\ W_{03}(s) &= |_{(w,*,c,*,\overline{B})\in s}\, L_3(w,c)\ |_{(w,e,c,i,*,B)\in s}\, R_3(w,c,e) \\ W_{04}(s) &= |_{(w,*,*,*,\overline{B})\in s}\, S_4(w)\ |_{(w,e,*,*,\overline{B})\in s} L_4(w,e))\ |_{(w,*,*,o,*,*)\in s}\, O_4(w) \end{aligned}$$

ここで、

[0099] [数27]

$\overline{B}$

は、B(ボタンを示す)を除く任意のタイプ(Bの補集合)を示し、*は何であってもよいことを示す。

基本構造:

画面sの基本構造SF(s)は、下式のように形式化される。

[0100] [数28]

$$SF(s) = W_{04}(s) \mid W_{03}(s) \mid W_{02}(s)$$

制御機能:

画面sに付随する制御機能は、下式のように形式化される。

[0101] [数29]

$$\Phi(s) = SF(s) \rhd [i^{s_k}?1].\Phi(s)$$

画面の集合に付随する制御機能は、下式のように形式化される。

[0102] [数30]

$$\Psi(s_1,\ldots,s_k) = (|_{s\in\{s_1,\ldots,s_k\}}\ [i^{s}?1.]\Phi(s)) \rhd [i^{s_0}?1].nil$$

s_0は、プログラムを終了したときの画面(プログラム自身の画面群には属さない終了画面)であるとする。

[0103] ここで制御機能の役割についてまとめると、下記のようになる。

[0104] [表11]

制御機能		役割
Ψ	プログラムに1つもうける	・自身は、エンドユーザによって起動され、終了ボタン押下による真の値を受信したとき終了する。 ・ユーザによるボタン押下などの指示による真の値を受信したとき、対応するΦを起動する。 ・終了ボタンの真の値を受信したとき、自身を終了する前に、全てのΦを終了させる。
Φ	SFごとに1つ	・自身はΨによって、起動および終了される。 ・指定チャネルで、ユーザによるボタン押下などの指示による真の信号を受信したときに、担当するSFを起動する。

本論では、Lyee計算法の実現方法の1つとして、SFは、画面に対して1つもうけ、1つのSFが、対応する画面に係わる入出力および計算の全てのプロセスを含むように形式化した。従ってSFの制御関数Φも、画面に1つである。しかし、画面に対して複数のSFおよび制御関数Φを設けるように形式化することも選択しとして可能である。実装方法は、プログラムとしての効率性によって決定すればよい。

Lyeeプログラム:

最後に、画面$s_1, \ldots, s_k$を含む要件に関するLyeeプログラム$P(s_1, \ldots, s_k)$は下式で表される。

[0105] [数31]

$$\mathcal{P}(s_1, \ldots, s_k) = \Psi(s_1, \ldots, s_k)/\mathcal{L}(s_1, \ldots, s_k)$$

ここで、$L(s_1, \ldots, s_k)$の集合は、環境との間のすべての入力チャネルおよび出力チャネルを含み、下式により定義される。

[0106] [数32]

$$\mathcal{L}(s_1, \ldots, s_k) = \left(\bigcup_{(w,*,*,i/o,*) \in s_k} \{d_w\} \right) \cup \{i^{s_1}\}$$

i/oの意味は、この項目は値i、oまたはioを含んでいなければならないことをさす。また、s_1は、ユーザがこのプログラムを実行した場合に最初に現れる画面である。

実施例 1

[0107] 5. ケーススタディ1

このセクションにおいては、Lyee計算法で単語を計算するために、Lyeeプログラムがどのように進行するかを、具体例を上げて段階的に説明する。

[0108] ここに示す例は、画面を1つしか持たない(2つの画面を持つ他の例は付録に記載した)。表12の要件に示すように、ユーザは、単語aを入力し、単語bの値を待ち、次に、ボタンB_0を押して画面を終了する。図11はこのような画面を示す。

[0109] [表12]

単語	定義式	実行条件	IO	タイプ
a			i	real
b	2*a	a>0	o	real
B_0	s_0	クリック	i	B

この画面s_1は3つの宣言からなる。

[0110] [数33]

$$s_1 = \{(a,,,i,real),(b,2*a,a>0,o,real),(B_0,s_0,Click,B)\}$$

前のセクションで説明した一般的なLyeeプログラムの定義によれば、表12の要件に関するLyeeプログラムは、下式により表される。

[0111] [数34]

$$SF(s_1) = W_{04}(s_1) \mid W_{03}(s_1) \mid W_{02}(s_1)$$

$$\Phi(s_1) = SF(s_1) \triangleright [i^{s_1}?1].\Phi(s_1)$$
$$\Psi(s_1) = ([i^{s_1}?1].\Phi(s_1)) \triangleright [i^{s_0}?1].nil$$
$$\mathcal{L}(s_1) = \{d_a, d_b, d_{B_0}, i^{s_1}\}$$

$$\mathcal{P}(s_1) = \Psi(s_1)/\mathcal{L}(s_1)$$

ここで、$W_{02}(s_1)$、$W_{03}(s_1)$および$W_{04}(s_1)$は、下記の通りである。前述したように、R_3のプロセスはLyee計算法では必須ではないので、省くことも可能である。

[0112] [表13]

$W_{02}(s_1) = L_2(a) \mid I_2(a)$	
$L_2(a) = [\imath_2^a?a].$ $[\jmath_4^a!a].$ nil	$I_2(a) = [d_a?a].$ $[\imath_2^a!a].$ nil

$W_{03}(s_1) = L_3(b, a>0) \mid R_3(B_0, click, s_0)$	
$L_3(b, a>0) = [\imath_4^a?a].$ $[\imath_3^b!(a>0)].$ nil	$R_3(B_0, click, s_0) = [d_{B_0}?click].$ $[\imath^{s_0}!1].$ nil

$W_{04}(s_1) = S_4(a) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b)$		
$S_4(a) = [\jmath_4^a?a].$ $S_4^a(a)$ $S_4(b) = [\jmath_4^b?b].$ $S_4^b(b)$	$L_4(b, 2*a) = [\imath_3^b?1].$ $[\imath_4^a?a].$ $[\jmath_4^b!(2*a)].$ nil	$O_4(b) = [\imath_4^b?b].$ $[d_b!b].$ nil

ここで、このプログラムP(S_1)のエンドユーザ(環境)が、下記の一連の行動を行いたいと考えていると仮定しよう。プログラムを実行し(画面s_1を起動し)、単語aに値「7」を与え、単語bの値を得るのを待ち、ボタンB_0を押してプログラムを終了する。このエンドユーザの行動は、下記のプロセスεにより捉えることができる。

[0113] [表14]

プロセス	コメント
$\mathcal{E} = [\imath^{s_1}!1].\mathcal{E}_1$	プログラム(画面s_1)を起動。
$\mathcal{E}_1 = [d_a!7].\mathcal{E}_2$	単語aに値「7」を与える。
$\mathcal{E}_2 = [d_b?x].\mathcal{E}_3$	単語bの値を得る。
$\mathcal{E}_3 = [d_{B_0}!click].nil$	終了するためにボタンB_0をクリック。

上記に示した、Lyee計算法によるプログラムP(S_1)のプロセスとエンドユーザのプロセスを図12に示した。

[0114] 次に、求められる目的に到達するために、プログラムP(S_1)がこの環境とどのように相互作用を行うかを見る。言い換えると、エンドユーザの行動を取り込むプログラムεと並行に実行されるプログラムP(S_1)の進行プロセスを追うのである。これは、すなわち、$P(s_1) \mid \varepsilon$のふるまいを探すことである。このプロセスのステップは下記の通りである

。定義式には、等号記号の左辺が右辺に置き換えられる根拠となる定義をコメントとして付した。また、プロセス間の相互作用によって、同期(synchronization)(同一チャネル上で値の送信と受信が同時に成立した)が起こった部分にもコメントを付した。下線部分が同期した送信および受信作用である。

[0115] [表15]

$\mathcal{P}(s_1) \mid \mathcal{E}$

$=$ 〈Pおよびεの定義により〉

$\Psi(s_1)/\mathcal{L}(s_1) \mid [i^{s_1}!1].\mathcal{E}_1$

$=$ 〈Ψの定義により〉

$(([i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1) \mid [i^{s_1}!1].\mathcal{E}_1$

$\xrightarrow{\tau}$ 〈同期：エンドユーザは、画面s_1を起動してプログラムを実行〉

$(\Phi(s_1) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1) \mid \mathcal{E}_1$

$=$ 〈Φとε_1の定義により〉

$((SF(s_1) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1) \mid [d_a!7].\mathcal{E}_2$

$=$ 〈SFの定義により〉

$(((W_{04}(s_1) \mid W_{02}(s_1) \mid W_{03}(s_1)) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1) \mid [d_a!7].\mathcal{E}_2$

$=$ 〈W_{02}およびその結果としてのI_2の定義により〉

$(((W_{04}(s_1) \mid L_2(a) \mid ([d_a?a].[i_2^a!a].nil) \mid W_{03}(s_1)) \rhd \dots) \rhd \dots)/\mathcal{L}(s_1) \mid [d_a!7].\mathcal{E}_2$

$\xrightarrow{\tau}$ 〈同期：I_2を通してエンドユーザが値「7」を単語aに与える〉

$(((W_{04}(s_1) \mid L_2(a) \mid ([i_2^a!7].nil) \mid W_{03}(s_1)) \rhd \dots) \rhd \dots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$=$ 〈L_2の定義により〉

$(((W_{04}(s_1) \mid ([i_2^a?a].[j_4^a!a].nil) \mid ([i_2^a!7].nil) \mid W_{03}(s_1)) \rhd \dots) \rhd \dots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$\xrightarrow{\tau}$ 〈同期：aの値7がL_2(a)に送られる〉

$(((W_{04}(s_1) \mid ([j_4^a!7].nil) \mid W_{03}(s_1)) \rhd \dots) \rhd \dots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$=$ 〈W_{04}の定義により〉

$((S_4(a) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid ([j_4^a!7].nil) \mid W_{03}(s_1)) \rhd \dots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$=$ 〈S_4(a)の定義により〉

$(((([j_4^a?a].S_4^a(a)) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid ([j_4^a!7].nil) \mid W_{03}(s_1)) \rhd \dots$

$\xrightarrow{\tau}$ 〈同期：aの値7がL_2(a)から送信され、メモリに記憶〉

$((S_4^a(7) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid W_{03}(s_1)) \rhd \dots$

$=$ 〈$S_4^a(7)$およびW_{03}の定義により〉

$(((([j_4^a?z].S_4^a(z) + [i_4^a!7].S_4^a(7)) \mid \dots \mid ((L_3(b, a>0) \mid R_3(B_0, click, s_0))) \rhd \dots$

$=$ 〈L_3の定義により〉

$(((([j_4^a?z].S_4^a(z) + [i_4^a!7].S_4^a(7)) \mid \dots \mid ((([i_4^a?a].[i_3^b!(a>0)].nil) \mid \dots$

$\xrightarrow{\tau}$ 〈同期：b条件を計算するために、aの値がL_3(b)に送られる〉

$((S_4^a(7) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid ([i_3^b!(7>0)].nil) \mid \dots$

$=$ 〈L_4の定義および（7＞0）の判定により〉

$((S_4^a(7) \mid S_4(b) \mid ([i_3^b?1].[i_4^a?a].[j_4^b!(2*a)].nil) \mid O_4(b) \mid ([i_3^b!1].nil) \mid \dots$

[0116] [表16]

$\xrightarrow{\tau}$ 〈同期：ｂの計算を実行するために、ｂ条件が判定され、
L_4(b)に送られる〉

$((S_4^a(7) \mid S_4(b) \mid ([i_4^a?a].[j_4^b!(2*a)].nil) \mid O_4(b) \mid \ldots$

$=$ 〈$S_4^a(7)$ の定義により〉

$(((\underline{[j_4^a?z]}.S_4^a(z) + [i_4^a!7].S_4^a(7)) \mid S_4(b) \mid (\underline{[i_4^a?a]}.[j_4^b!(2*a)].nil) \mid O_4(b) \mid \ldots$

$\xrightarrow{\tau}$ 〈同期：ａの値がL_4に送られる〉

$((S_4^a(7) \mid S_4(b) \mid ([j_4^b!(2*7)].nil) \mid O_4(b) \mid R_3(B_0, click, s_0)) \rhd \ldots$

$=$ 〈S_4(b)の定義および（２＊７）の計算により〉

$((S_4^a(7) \mid (\underline{[j_4^b?b]}.S_4^b(b)) \mid (\underline{[j_4^b!(14)]}.nil) \mid O_4(b) \mid R_3(B_0, click, s_0)) \rhd \ldots$

$\xrightarrow{\tau}$ 〈同期：ｂの値１４がL_4(b)から送信され、の値をメモリに記憶〉

$((S_4^a(7) \mid \underline{S_4^b(14)} \mid O_4(b) \mid R_3(B_0, click, s_0)) \rhd \ldots$

$=$ 〈$S_4^b(14)$ およびO_4（ｂ）の定義により〉

$((S_4^a(7) \mid ([j_4^b?z].S_4^b(z) + \underline{[i_4^b!14]}.S_4^b(14)) \mid (\underline{[i_4^b?b]}.[d_b!b].nil) \mid R_3(B_0, click, s_0)) \rhd \ldots$

$\xrightarrow{\tau}$ 〈同期：ｂの値１４がO_4に送られる〉

$((S_4^a(7) \mid S_4^b(14) \mid ([d_b!14].nil) \mid R_3(B_0, click, s_0)) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$=$ 〈ε_2の定義により〉

$((S_4^a(7) \mid S_4^b(14) \mid (\underline{[d_b!14]}.nil) \mid R_3(B_0, click, s_0)) \rhd \ldots)/\mathcal{L}(s_1) \mid \underline{[d_b?x]}.\mathcal{E}_3$

$\xrightarrow{\tau}$ 〈同期：O_4内に記憶されているｂの値が環境に送られる〉

$((S_4^a(7) \mid S_4^b(14) \mid R_3(B_0, click, s_0)) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_3$

$=$ 〈R_3およびε_3の定義により〉

$((S_4^a(7) \mid S_4^b(14) \mid (\underline{[d_{B_0}?click]}.[i^{s_0}!1].nil)) \rhd \ldots)/\mathcal{L}(s_1) \mid \underline{[d_{B_0}!click]}.nil$

$\xrightarrow{\tau}$ 〈同期：エンドユーザが終了のためにボタンB_0を押す〉

$((S_4^a(7) \mid S_4^b(14) \mid (\underline{[i^{s_0}!1]}.nil)) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd \underline{[i^{s_0}?1]}.nil)/\mathcal{L}(s_1)$

$\xrightarrow{\tau}$ 〈同期：システムがその実行を終了〉

nil

まとめると、プログラムの進行の主要なステップは下記の通りである。

エンドユーザが、プログラム起動のコマンドを送ることによって、制御関数$\Psi(s_1)$が起動されて、制御関数$\Psi(s_1)$が制御関数$\Phi(s_1)$を起動し、制御関数(s_1)が初期画面s_1とSF(s_1)を起動することによって、プログラムが起動する。

[0117] [数35]

$((\underline{[i^{s_1}?1]}.\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1) \mid \underline{[i^{s_1}!1]}.\mathcal{E}_1$

エンドユーザが単語aに値「7」を与える(送信する)と、SF(s_1)の中のI_2(a)が値7を受信する。

[0118] [数36]

$$(((W_{04}(s_1) \mid L_2(a) \mid (\underline{[d_a?a]}.[i_2^a!a].nil) \mid W_{03}(s_1)) \rhd \ldots) \rhd \ldots)/\mathcal{L}(s_1) \mid \underline{[d_a!7]}.\mathcal{E}_2$$

I_2(a)は、aの値7をL_2(a)に送信する。

[0119] [数37]

$$(((W_{04}(s_1) \mid (\underline{[i_2^a?a]}.[j_4^a!a].nil) \mid (\underline{[i_2^a!7]}.nil) \mid W_{03}(s_1)) \rhd \ldots) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$$

L_2(a)が送信したaの値7が、メモリ、すなわち、S_4(a)に記憶される(S4(a)に初期値が記憶されることによって初期化された)。

[0120] [数38]

$$(((\underline{[j_4^a?a]}.S_4^a(a)) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid (\underline{[j_4^a!7]}.nil) \mid W_{03}(s_1)) \rhd \ldots$$

bの計算条件を計算するために、aの値7が、初期化されたaのメモリS_4からL_3(b, a>0)に送られる。

[0121] [数39]

$$(((\underline{[j_4^a?z]}.S_4^a(z) + \underline{[i_4^a!7]}.S_4^a(7)) \mid \ldots \mid ((\underline{[i_4^a?a]}.[i_3^b!(a>0)].nil) \mid \ldots$$

bの計算条件が判定され(a＞0は真(1)と判定)、L_4(b, 2*a)に値1が送られる。

[0122] [数40]

$$((S_4^a(7) \mid S_4(b) \mid (\underline{[i_3^b?1]}.[i_4^a?a].[j_4^b!(2*a)].nil) \mid O_4(b) \mid (\underline{[i_3^b!1]}.nil) \mid \ldots$$

aの値7が、値7を保持しているaのメモリS_4からL_4(b, 2*a)に送られる。

[0123] [数41]

$$(((\underline{[j_4^a?z]}.S_4^a(z) + \underline{[i_4^a!7]}.S_4^a(7)) \mid S_4(b) \mid (\underline{[i_4^a?a]}.[j_4^b!(2*a)].nil) \mid O_4(b) \mid \ldots$$

bの定義式「2*a」が計算され、bの値14が、メモリ、すなわち、S_4(b)に記憶される(S_4(b)が初期化された)。

[0124] [数42]

$$((S_4^a(7) \mid (\underline{[j_4^b?b]}.S_4^b(b)) \mid (\underline{[j_4^b!(14)]}.nil) \mid O_4(b) \mid R_3(B_0, click, s_0)) \rhd \ldots$$

bの値14が、bのメモリS_4から、出力プロセスO_4(b)に送られる。

[0125] [数43]

$$\left((S_4^a(7) \mid ([\underline{\jmath_4^b?z}].S_4^b(z) + [\underline{\imath_4^b!14}].S_4^b(14)) \mid ([\underline{\imath_4^b?b}].[d_b!b].nil) \mid R_3(B_0, click, s_0) \right) \rhd \ldots$$

10. O_4によってbの値が環境に送られ、エンドユーザはbの値を受信する。この時点で起動中のSF(s_1)に属するプロセスは、値を保持しているaおよびbのS_4と、R3(B_0, click, s_0)である。

[0126] [数44]

$$\left((S_4^a(7) \mid S_4^b(14) \mid ([\underline{d_b!14}].nil) \mid R_3(B_0, click, s_0)) \rhd \ldots \right) / \mathcal{L}(s_1) \mid [\underline{d_b?x}].\mathcal{E}_3$$

11. エンドユーザが、プログラム終了のためにボタンB_0を押し、R3(B_0, click, s_0)は値clickを受信、自身の計算条件を判定して真の値1を制御関数Ψ(s_1)に送信する。真の値1を受信したΨ(s_1)は、自身が終了する前にΦ(s_1)を終了させ、Φ(s_1)は自身が終了する前にSF(s_1)を終了させる。

[0127] [数45]

$$\left((S_4^a(7) \mid S_4^b(14) \mid ([\underline{d_{B_0}?click}].[\imath^{s_0}!1].nil)) \rhd \ldots \right) / \mathcal{L}(s_1) \mid [\underline{d_{B_0}!click}].nil$$

図13は、これらの各ステップを示す。

[0128] かっこ付き番号は、上記のステップの説明に対応する。かっこなしの数字は、送受信される値を示している。

[0129] さらに、各プロセスの遷移は証明によって確認することができる。例えば、下記のプロセスの遷移がある(上記ステップ1に対応)。P(s_1)とεの間に同期が起こり(プログラム起動コマンドの送受信)、P_1(s_1)とε_1の並行プロセスに移行するプロセス遷移を表わしている。

[0130] [数46]

$$(\mathcal{P}(s_1) \mid \mathcal{E}) \xrightarrow{\tau} (\mathcal{P}_1(s_1) \mid \mathcal{E}_1)$$

このとき、

[0131] [数47]

$$\mathcal{P}_1(s_1) = (\Phi(s_1) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1)$$

である。

上記は、表5の規則を用いて、次の証明で正しいことが確認できる。

[0132] [数48]

$$R_{?}:\frac{1\in[\![1]\!]}{[i^{s_1}?1].\Phi(s_1)\ \overset{i^{s_1}?1}{\rightarrow}\ \Phi(s_1)}\qquad R_{!}:\frac{1\in[\![1]\!]}{\mathcal{E}\ \overset{i^{s_1}!1}{\rightarrow}\ \mathcal{E}_1}$$

$$R^{l}_{\triangleright}:\frac{[i^{s_1}?1].\Phi(s_1)\ \overset{i^{s_1}?1}{\rightarrow}\ \Phi(s_1)}{\Psi(s_1)\ \overset{i^{s_1}?1}{\rightarrow}\ \Phi(s_1)\triangleright[i^{s_0}?1].nil}$$

$$R_{/}:\frac{\Psi(s_1)\ \overset{i^{s_1}?1}{\rightarrow}\ \Phi(s_1)\triangleright[i^{s_0}?1].nil}{\mathcal{P}(s_1)\ \overset{i^{s_1}?1}{\rightarrow}\ \mathcal{P}_1(s_1)}$$

$$R^{\tau}_{|}:\frac{\mathcal{P}(s_1)\ \overset{i^{s_1}?1}{\rightarrow}\ \mathcal{P}_1(s_1)\qquad \mathcal{E}\ \overset{i^{s_1}!1}{\rightarrow}\ \mathcal{E}_1}{(\mathcal{P}(s_1)\mid\mathcal{E})\ \overset{\tau}{\rightarrow}\ (\mathcal{P}_1(s_1)\mid\mathcal{E}_1)}$$

6. 結論

本稿は、最初に、Lyee方法論の基本概念を容易かつ本質的にサポートするLyee計算法と呼ぶ形式的プロセス代数を定義した。実際に、この計算法は、フォン・ノイマンのものより、Lyee方法論の概念をサポートするのにより適した抽象機械と見なすことができる。この機械は、プログラムを最終結果を生成するために共に相互作用し合う分子の集合と見なす。二番目に、この計算法がどのようにして、全ソフトウェア生成プロセスとLyeeベクトルの両方を形式化し、簡易化することができるのかを示した。実際、経路作用要素(routing vector)、はもはや必要としない。また、パレット関数も不要となり、制御機能の役割は大幅に簡易化される。Lyeeソフトウェア生成の自動生成の全プロセスを形式化した。この形式化によって、この方法論の意味論を、この方法論の背後にある概念を明快に理解することができるように形式化することができる。その上、この形式的記述は、いろいろな局面におけるこの方法論の多くの興味ある分析を始めるためには不可避である。例えば、Lyeeで生成したソフトウェアまたはソフトウェア生成プロセスを最適化するためには、最適化されたプログラムがその元のバージョンと同じものである、という形式的証明を必要とする。プロセス代数を使用すれば、形式的に同じものであることのチェック、またはもっと一般的にモデル・チェックをうまく行うことができる。

[0133] 将来の研究として、我々はLyee方法論の意味論をさらに研究し、Lyee方法論をより簡素化し、要件からより信頼性の高い最適化されたコードの生成が行えるようにしたい。最適化するために、Lyee計算法のプロセス間の適合(congruence)関係を定義し、それによってすべてのそこから帰結する最適化が正しいことを証明したいと考えている。

実施例 2

[0134] ＜付録＞

ケーススタディ2—画面が2つの場合

複数画面間の相互作用を示すために、2つの画面からなるプログラムのケースについて述べる。図14がそのプログラムの画面を示した図である。このプログラムを起動すると画面S_1が表示される。画面S_1で、ユーザは、単語aを入力し、単語bの値が出力されるのを待つ。ボタンB_2を押すと、画面S_2が表示され、単語bの値が出力されるのを待つ。画面S_2でボタンB_0をおすと、プログラムを終了する。

[0135] 図14のプログラムの要件の宣言をまとめると、画面S_1の要件は表17、画面S_2の要件は表18のようになる。

[0136] [表17]

単語	定義式	実行条件	IO	...
a			I	...
b	2*a	a>0	O	...
B_2	s_2	クリック	I	...

[0137] [表18]

単語	定義式	実行条件	IO	...
e	1+b	b>0	O	...
B_0	s_0	クリック	I	...

前のセクションで説明した一般的なLyeeプログラムの定義によれば、表17および表18の要件に関するLyeeプログラムは、下式により表される。

[0138] [数49]

$$SF(s_1) = W_{04}(s_1) \mid W_{03}(s_1) \mid W_{02}(s_1)$$
$$SF(s_2) = W_{04}(s_1) \mid W_{03}(s_1)$$

画面s_2に対するW_{02}が存在しないことに留意されたい。何故なら、W_{02}は、ボタン以外に入力単語を含んでいないからである。

[0139] [数50]

$$\Phi(s_1) = SF(s_1) \rhd [i^{s_1}?1].\Phi(s_1)$$
$$\Phi(s_2) = SF(s_2) \rhd [i^{s_2}?1].\Phi(s_2)$$
$$\Psi(s_1, s_2) = ([i^{s_1}?1].\Phi(s_1) \mid [i^{s_2}?1].\Phi(s_2)) \rhd [i^{s_0}?1].nil$$
$$\mathcal{L}(s_1, s_2) = \{d_a, d_b, d_e, d_{B_0}, d_{B_2}\}$$
$$\mathcal{P}(s_1, s_2) = \Psi(s_1, s_2)/\mathcal{L}(s_1, s_2)$$

ここにおいては、$W_{02}(s_1)$、$W_{03}(s_1)$、$W_{04}(s_1)$および、$W_{03}(s_2)$、$W_{04}(s_2)$は、以下のように定義される。ケーススタディ1と同様に、R_3のプロセスはLyee計算法では必須ではないので、省くことも可能である。

[0140] [表19]

$W_{02}(s_1) = L_2(a) \mid I_2(a)$	
$L_2(a) = [i_2^a?a].$ $[i_4^a!a].$ nil	$I_2(a) = [d_a?a].$ $[i_2^a!a].$ nil

[0141] [表20]

$W_{03}(s_1) = L_3(b, a > 0) \mid R_3(B_2, click, s_2)$	
$L_3(b, a > 0) = [i_4^a?a].$ $[i_3^b!(a > 0)].$ nil	$R_3(B_2, click, s_2) = [d_{B_2}?click].$ $[i^{s_2}!1].$ nil
$W_{03}(s_2) = L_3(e, b > 0) \mid R_3(B_0, click, s_0)$	
$L_3(e, b > 0) = [i_4^b?b].$ $[i_3^e!(b > 0)].$ nil	$R_3(B_0, click, s_0) = [d_{B_0}?click].$ $[i^{s_0}!1].$ nil

[0142] [表21]

$W_{04}(s_1) = S_4(a) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b)$		
$S_4(a) = [\jmath_4^a?a].S_4^a(y)$ $S_4(b) = [\jmath_4^b?b].S_4^b(y)$	$L_4(b, 2*a) = [\imath_3^b!1].[\imath_4^a?a].[\jmath_4^b!(2*a)].nil$	$O_4(b) = [\imath_4^b?b].[d_b!b].nil$
$W_{04}(s_2) = S_4(e) \mid L_4(e, 1+b) \mid O_4(e)$		
$S_4(e) = [\jmath_4^e?e].S_4^e(y)$	$L_4(e, 1+b) = [\imath_3^e!1].[\imath_4^b?b].[\jmath_4^e!(1+b)].nil$	$O_4(e) = [\imath_4^e?e].[d_e!e].nil$

ここで、このプログラムのエンドユーザ(環境)が、下記の一連の行動を行いたいと考えているとする。プログラムを実行し(画面s_1を起動し)、単語aに値「7」を与え、単語bの値を得るために待機し、画面s_2に行くためにボタンB_2を押し、値eを得るために待機し、ボタンB_0を押すことによりプログラムを終了する。この行動は、下記のプロセスεにより捉えることができる。

[0143] [表22]

エンドユーザの行動	コメント
$\mathcal{E} = [\imath^{s_1}!1].\mathcal{E}_1$	プログラムを起動。
$\mathcal{E}_1 = \overline{[d_a!7].\mathcal{E}_2}$	単語aに値7を付与。
$\mathcal{E}_2 = \overline{[d_b?b].\mathcal{E}_3}$	単語bの値を得るために待機。
$\mathcal{E}_3 = [\imath^{B_2}!click].\mathcal{E}_4$	画面2に行くためにボタンB_2を押す。
$\mathcal{E}_4 = [d_e?e].\mathcal{E}_5$	単語eの値を入手するために待機。
$\mathcal{E}_5 = [\imath^{B_0}!click].nil$	終了のためにボタンB_0を押す。

上記に示した、Lyee計算法によるプログラムP(S_1)のプロセスとエンドユーザのプロセスを図15に示した。

[0144] プログラムの進行の主要なステップは下記の通りである。

エンドユーザが、プログラム起動のコマンドを送ることによって、制御関数Ψ(s_1, s_2)が起動されて、制御関数Ψ(s_1, s_2)が制御関数Φ(s_1)を起動し、制御関数(s_1)が初期画面s_1とSF(s_1)を起動することによって、プログラムが起動する。

P(s_1, s_2)とεの間に同期が起こり(プログラム起動コマンドの送受信)、P_1(s_1, s_2)とε$_1$の並行プロセスに移行。

[0145] [数51]

$$\mathcal{P}(s_1, s_2) \mid \mathcal{E} \xrightarrow{\tau} \mathcal{P}_1(s_1, s_2) \mid \mathcal{E}_1$$

ここで、

[0146] [数52]

$$\mathcal{P}_1(s_1, s_2) = (\Phi(s_1) \mid [i^{s_2}?1].\Phi(s_2) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

エンドユーザが単語aに値「7」を与える(送信する)と、SF(s_1)の中のI_2(a)が値7を受信する。

P_1(s_1, s_2)とε_1の間に同期が起こり(単語aの値の送受信)、P_2(s_1, s_2)とε_2の並行プロセスに移行。

[0147] [数53]

$$\mathcal{P}_1(s_1, s_2) \mid \mathcal{E}_1 \xrightarrow{\tau} \mathcal{P}_2(s_1, s_2) \mid \mathcal{E}_2$$

ここで、

[0148] [数54]

$$\begin{aligned}\mathcal{P}_2(s_1, s_2) \quad = \quad & ((((W_{04}(s_1) \mid [i_2^a!7].nil \mid L_2(a) \mid W_{03}(s_1)) \rhd [i^{s_1}?1].\Phi(s_1) \mid [i^{s_2}?1].\Phi(s_2)) \\ & \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)\end{aligned}$$

aの値7がメモリS_4(a)に記憶される。この記憶は、2つのステップにより行うことができる。第1のステップにおいて、aの値7がL_2(a)に送られる。第2のステップにおいて、L_2(a)は、この値をメモリS_4(a)に送る。

第1のステップは下記の通りである。P_2(s_1, s_2)の中で同期が起こり(I_2(a)とL_2(a)間の単語aの値7の送受信)、P_3(s_1, s_2)とε_2の並行プロセスに移行。

:

[0149] [数55]

$$\mathcal{P}_2(s_1, s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_3(s_1, s_2) \mid \mathcal{E}_2$$

ここで、

[0150] [数56]

$$\mathcal{P}_3(s_1,s_2) = (((W_{04}(s_1) \mid [\gamma_4^a!7].nil \mid W_{03}(s_1)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

第2のステップにおいて、aの値7はW_{04}のメモリS_4(a)に記憶される(S_4(a)に初期値が記憶されることによって初期化された)。P_3(s_1, s_2)の中で同期が起こり(L_2(a)とS_4(a)間の単語aの値7の送受信)、P_4(s_1, s_2)とε_2の並行プロセスに移行。

:

[0151] [数57]

$$\mathcal{P}_3(s_1,s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_4(s_1,s_2) \mid \mathcal{E}_2$$

ここで、

[0152] [数58]

$$\mathcal{P}_4(s_1,s_2) = (((S_4^a(7) \mid S_4(b) \mid L_4(b,2*a) \mid O_4(b) \mid W_{03}(s_1)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

bの計算条件を計算するために、aの値7が、初期化されたaのメモリS_4からL_3(b, a>0)に送られる。P_4(s_1, s_2)の中で同期が起こり(S_4(a)とL_3(b, a>0)の間の送受信)、P_5(s_1, s_2)とε_2の並行プロセスに移行。:

[0153] [数59]

$$\mathcal{P}_4(s_1,s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_5(s_1,s_2) \mid \mathcal{E}_2$$

ここで、

[0154] [数60]

$$\mathcal{P}_5(s_1,s_2) = (((S_4^a(7) \mid S_4(b) \mid L_4(b,2*a) \mid O_4(b) \mid [\imath_3^b!(7>0)].nil \mid R_3(B_2,s_2,click)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

bの計算条件が判定され(a＞0は真(1)と判定)、L_4(b, 2*a)に値1が送られる。P_5(s_1, s_2)の中で同期が起こり(L_3(b, a>0)とL_4(b, 2*a)の間の真偽値1の送受信)、P_6(s_1, s_2)とε_2の並行プロセスに移行。:

[0155] [数61]

$$\mathcal{P}_5(s_1,s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_6(s_1,s_2) \mid \mathcal{E}_2$$

ここで、

[0156] [数62]

$$\mathcal{P}_6(s_1,s_2) = (((S_4^a(7) \mid S_4(b) \mid [j_4^a!(2*a)].nil \mid O_4(b) \mid R_3(B_2,s_2,click)) \triangleright [i^{s_1}?1].\Phi(s_1) \mid [i^{s_2}?1].\Phi(s_2)) \triangleright [i^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

aの値7がaのメモリS_4からL_4(b, 2*a)に送られ、bの定義式「2*a」が計算され、bの値14が、メモリS_4(b)に記憶される(S_4(b)が初期化された)。P_6(s_1, s_2)の中で同期が起こり(aのメモリS_4とL_4(b, 2*a)との間、L_4(b, 2*a)とbのメモリS_4の間の送受信)、P_7(s_1, s_2)とε$_2$の並行プロセスに移行。:

[0157] [数63]

$$\mathcal{P}_6(s_1,s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_7(s_1,s_2) \mid \mathcal{E}_2$$

ここで、

[0158] [数64]

$$\mathcal{P}_7(s_1,s_2) = (((S_4^a(7) \mid S_4^b(14) \mid O_4(b)) \triangleright [i^{s_1}?1].nil \mid R_3(B_2,s_2,click)) \triangleright [i^{s_1}?1].\Phi(s_1) \mid [i^{s_2}?1].\Phi(s_2)) \triangleright [i^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

bの値14が、bのメモリS_4から、出力プロセスO_4(b)に送られる。O_4(b)によってbの値が環境に送られ、エンドユーザはbの値を受信する。P_7(s_1, s_2)の中の同期(L_4(b, 2*a)とO_4(b)の間の送受信)と、P_7(s_1, s_2)とε$_2$間の同期(単語bの値の送受信)が起こり、P_8(s_1, s_2)とε$_3$の並行プロセスに移行。

この時点で起動中のSF(s_1)に属するプロセスは、値を保持しているaおよびbのS_4と、R3(B_0, click, s_0)である。:

[0159] [数65]

$$\mathcal{P}_7(s_1,s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_8(s_1,s_2) \mid \mathcal{E}_3$$

ここで、

[0160] [数66]

$$\mathcal{P}_8(s_1,s_2) = (((S_4^a(7) \mid S_4^b(14) \mid R_3(B_2,s_2,click)) \triangleright [i^{s_1}?1].\Phi(s_1) \mid [i^{s_2}?1].\Phi(s_2)) \triangleright [i^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

画面2に行くために、エンドユーザは画面s_1上のボタンB_2を押す。ε$_3$から送信された

ボタンB_2のClick値をR3(B_2, click, s_2)が受信、R3(B_2, click, s_2)は自身の計算条件を判定して真の値1を制御関数Ψ(s_1, s_2)に送信する。値1を受信した制御関数Ψ(s_1, s_2)は制御関数Φ(s_2)を起動。値1を受信した制御関数Φ(s_2)は、SF(s_2)を起動し、画面s_2を起動する。

P_8(s_1, s_2)とε_3の間の同期(ボタンB_2のclick値の送受信と)、P_8(s_1, s_2)の中の同期(R3(B_2, click, s_2)と、Ψ(s_1, s_2)および制御関数Φ(s_2)間の送受信)が起こり、P_9(s_1, s_2)とε_4の並行プロセスに移行。:

[0161] [数67]

$$\mathcal{P}_8(s_1, s_2) \mid \mathcal{E}_3 \xrightarrow{\tau} \mathcal{P}_9(s_1, s_2) \mid \mathcal{E}_4$$

ここで、

[0162] [数68]

$$\mathcal{P}_9(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14)) \rhd [i^{s_1}?1].\Phi(s_1) \mid \Phi(s_2)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

eの条件を計算するために、bのメモリS_4から値14をL_3(e, b>0)に送信する。P_9(s_1, s_2)の中で同期が起こり(aのメモリS_4とL_3(e, b>0)との間の送受信)、P_{10}(s_1, s_2)とε_4の並行プロセスに移行。:

[0163] [数69]

$$\mathcal{P}_9(s_1, s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{10}(s_1, s_2) \mid \mathcal{E}_4$$

ここで、

[0164] [数70]

$$\begin{aligned}\mathcal{P}_{10}(s_1, s_2) \quad = \quad & (((S_4^a(7) \mid S_4^b(14)) \rhd [i^{s_1}?1].\Phi(s_1) \mid (W_{04}(s_2) \mid [i_3^e(14 > 0)].nil \mid \\ & R_3(B_0, s_0, click)) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)\end{aligned}$$

eの計算条件b＞0が真(1)と判定され、L_3(e, b>0)から真の値1がL_4(e, 1+b)に送信され、eの値が計算される。P_{10}(s_1, s_2)の中で同期が起こり(L_3(e, b>0)とL_4(e, 1+b)の間の送受信)、P_{11}(s_1, s_2)とε_4の並行プロセスに移行。:

[0165] [数71]

$$\mathcal{P}_{10}(s_1, s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{11}(s_1, s_2) \mid \mathcal{E}_4$$

ここで、

[0166] [数72]

$$\mathcal{P}_{11}(s_1,s_2) = (((S_4^a(7) \mid S_4^b(14)) \rhd [i^{s_1}?1].\Phi(s_1) \mid (S_4(e) \mid [j_4^e(1+14)].nil \mid O_4(e) \mid R_3(B_0,s_0,click)) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

eの値を、L_4(e, 1+b)から、eのメモリS_4(b)に記憶する(S_4(b)が初期化された)。P_{11}(s_1, s_2)の中で同期が起こり(L_4(e)とeのメモリS_4の間の送受信)、P_{11}(s_1, s_2)とε_4の並行プロセスに移行。:。:

[0167] [数73]

$$\mathcal{P}_{11}(s_1,s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{12}(s_1,s_2) \mid \mathcal{E}_4$$

ここで、

[0168] [数74]

$$\mathcal{P}_{12}(s_1,s_2) = (((S_4^a(7) \mid S_4^b(14)) \rhd [i^{s_1}?1].\Phi(s_1) \mid (S_4^e(15) \mid O_4(e) \mid R_3(B_0,s_0,click)) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

eのメモリS_4(b)からeの値15を、O_4(e)に送信、O_4(e)からエンドユーザε_4にeの値を出力する。P_{11}(s_1, s_2)とε_4の間で同期が起こり(単語eの値14の送受信)、P_{12}(s_1, s_2)とε_5の並行プロセスに移行。:

[0169] [数75]

$$\mathcal{P}_{11}(s_1,s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{12}(s_1,s_2) \mid \mathcal{E}_5$$

ここで、

[0170] [数76]

$$\mathcal{P}_{12}(s_1,s_2) = (((S_4^a(7) \mid S_4^b(14)) \rhd [i^{s_1}?1].\Phi(s_1) \mid (S_4^e(15) \mid R_3(B_0,s_0,click)) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$

エンドユーザが、プログラムを終了するためにボタンB_0を押す。R3(B_0, click, s_0)は、エンドユーザε_5らボタンB_0の値clickを受信し、制御関数Ψ(s_1, s_2)に送信する。Ψ(s_1, s_2)は、自身が終了する前にΦ(s_1)およびΦ(s_2)を終了させ、Φ(s_1)およびΦ(s_2)は自身が終了する前に、それぞれSF(s_1)およびSF(s_2)を終了させる。P_{12}(s_1, s_2)と

ε_5の間で同期が起こり(CClick値の送受信)、次に$P_{12}(s_1, s_2)$の中で同期(R3(B_0, click, s_0)と、$\Psi(s_1, s_2)$、制御関数$\Phi(s_1)$および$\Phi(s_2)$間の送受信)が起こり、すべてのプロセスがnilに移行する。:

[0171] [数77]

$$\begin{aligned}\mathcal{P}_{12}(s_1,s_2) \mid \mathcal{E}_5 &\xrightarrow{\tau} (((S_4^a(7) \mid S_4^b(14)) \rhd [i^{s_1}?1].\Phi(s_1) \mid (S_4^c(15) \mid \\ &\quad [i^{s_0}?1].nil) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1,s_2) \\ &\xrightarrow{\tau} nil\end{aligned}$$

なお、本発明は、上述した実施形態および実施例には限定されず、本発明の技術思想の範囲内で様々な変形が可能である。たとえば、ビジネス・メソッド、ソフトウェア開発装置、ソフトウェア開発支援装置、ソフトウェア開発管理装置、あるいはこれらの機能をコンピュータに実現するためのソフトウェア並びに該ソフトウェアを搭載した記録媒体・専用機、などとしてもそれぞれ実現することが可能である。さらに、上記で説明したように本発明は、方法としても、或いはかかる機能を備えるソフトウェアとしても、該ソフトウェアが搭載される装置・ツール(ソフトウェア自体の場合も含む)としても、さらにはシステムとしても、実現され得るのはもとよりである。

[0172] 例えば図16は、本発明の異なる一実施形態として、「開発対象のソフトウェア」を生産するためのプログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開発支援装置、或いはソフトウェア開発管理装置のいずれかとして本発明を実施する場合に機能として備える構成を示した機能ブロック図である。同図に示すように、本発明に係るプログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開発支援装置、或いはソフトウェア開発管理装置は、全体制御部1601、宣言情報挿入部1602、制御関数Φ配置部1603、制御関数Ψ配置部1604及び情報記憶部1605を備えて構成される。

[0173] 全体制御部1601は、本プログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開発支援装置、或いはソフトウェア開発管理装置の全体の動

作制御、タイミング制御、入出力制御等を行う機能を有しており、当該機能を持つ専用チップ、専用回路、または当該機能をコンピュータに果たさせるためのソフトウェア(ツールとしてのソフトウェアも含む)、或いは該ソフトウェアを記録した記録媒体、当該記録媒体を搭載した処理装置・管理装置・ツールとして実現される。

[0174] 宣言情報挿入部1602は、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L2、L3、L4)および、作用要素(I2、O4、S4)のひな型の未定義部分に、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を挿入するための機能を有しており、当該機能を持つ専用チップ、専用回路、または当該機能をコンピュータに果たさせるためのソフトウェア(ツールとしてのソフトウェアも含む)、或いは該ソフトウェアを記録した記録媒体、当該記録媒体を搭載した処理装置・管理装置・ツールとして実現される。

[0175] 制御関数Φ配置部1603は、前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦが、制御するように関連づけるための機能を有しており、当該機能を持つ専用チップ、専用回路、または当該機能をコンピュータに果たさせるためのソフトウェア(ツールとしてのソフトウェアも含む)、或いは該ソフトウェアを記録した記録媒体、当該記録媒体を搭載した処理装置・管理装置・ツールとして実現される。

[0176] 制御関数Ψ配置部1604は、1つの制御関数モジュールΨを、前記制御関数Φを制御するように前記制御関数Φに関連づけるための機能を有しており、当該機能を持つ専用チップ、専用回路、または当該機能をコンピュータに果たさせるためのソフトウェア(ツールとしてのソフトウェアも含む)、或いは該ソフトウェアを記録した記録媒体、当該記録媒体を搭載した処理装置・管理装置・ツールとして実現される。

[0177] 情報記憶部1605は、プログラム情報のほか、各種制御情報、プロセス・セルとしてモジュール化された論理要素(L2、L3、L4)および、作用要素(I2、O4、S4)の前記ひな型、前記制御関数モジュールΦのひな型、前記制御関数モジュールΨのひな型、目的プログラム等の格納のほか、一時記憶のためのメモリとしても用いられる。

[0178] 図17は、上記の構成を備えるプログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開発支援装置、或いはソフトウェア開発管理装置のいずれかとして実施される本発明の動作を示したフローチャートである。

[0179] 同図に示すように、まず、宣言情報挿入部1602は、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L2、L3、L4)および、作用要素(I2、O4、S4)のひな型の未定義部分に、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を挿入する。(ステップ1701)。

[0180] すると、制御関数Φ配置部1603は、前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦが、制御するように関連づける。(ステップ1702)。

[0181] 次に、制御関数Ψ配置部1604は、1つの制御関数モジュールΨを、前記制御関数Φを制御するように前記制御関数Φに関連づける。(ステップ1703)。

[0182] よって、上記のような構成を備える本発明によれば、本発明の独自の体系に基づいて要求定義をとらえ、それを本発明独自の構造を備えるプロセス・セルとしてモジュール化され論理要素(L2、L3、L4)および、作用要素(I2、O4、S4)のひな型の未定義部分に代入するので、所望のソフトウェアが、属人性なく得られる。

[0183] 本発明の異なる実施体としての「開発対象のソフトウェア」を生産するためのプログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開発支援装置、ソフトウェア開発管理装置は、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型の未定義部分に、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を挿入する、宣言情報挿入手段と、前記論理要素と作用要素を、同一画面からのコマンドで相互作用

を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦを、前記単位を制御するように関連づける、制御関数Φ配置手段と、1つの制御関数モジュールΨを、前記制御関数Φを制御するように前記制御関数Φに関連づける、制御関数Ψ配置手段とを具備するように構成することもできる。

[0184] 本発明はさらに、上述の「開発対象のソフトウェアを生産する方法」によって生産されたソフトウェア、及び当該ソフトウェアが搭載された記録媒体或いは当該ソフトウェアが搭載された装置(ハードウェア)としても実現されるが、この場合の本発明は、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型の未定義部分に挿入したモジュール群と、前記モジュール群を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦで制御するように関連づけている1つまたは複数の制御関数モジュールΦと、前記制御関数Φを、1つの制御関数モジュールΨで制御するように関連づけている制御関数モジュールΨで構成されることもできる。

[0185] またさらに本発明は、上述の「開発対象のソフトウェアを生産する方法」によってソフトウェアを生産するために用いられるソフトウェアコードの雛型としてのソフトウェア、及び当該ソフトウェアが搭載された記録媒体或いは当該ソフトウェアが搭載された装置(ハードウェア)としても実現されるが、この場合の本発明は、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報を、埋め込むべき未定義部分を有した、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型と、前記宣言の情報を未定義部分に挿入した前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦで制御するための機能を有した制御関数モジュールΦのひな型と、前記制御関数Φを1つの制御関数モジュールΨで制御する

ための機能を有した制御関数モジュールΨのひな型とを備えるソフトウェアとしてコード化可能なひな型として実現してもよい。

[0186] さらに、本発明は、上述の「開発対象のソフトウェアを生産する方法」による、要件から抽出した情報(ドキュメント(紙、データ))の抽出方法として、またかかる抽出方法によって抽出された情報(ドキュメント(紙、データ))として、さらには当該抽出された情報の使用方法として、或いは、これらの情報が搭載された情報記録媒体として、または情報の抽出方法／使用方法がコード化されたソフトウェア、当該ソフトウェアが搭載された記録媒体／装置(ハードウェア)として、いずれも実現することができるが、この場合の本発明は、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)のひな型の未定義部分に挿入すべき、1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとの、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性からなる宣言として情報化した情報と、前記宣言の情報を未定義部分に挿入した前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、1つの制御関数モジュールΦで制御するように関連づけるための情報と、前記制御関数Φを1つの制御関数モジュールΨで制御するように関連づける情報とを備えるソフトウェア開発要件から抽出した情報として実現してもよい。

[0187] さらに本願発明は、その技術思想の同一及び等価に及ぶ範囲において様々な変形、追加、置換、拡大、縮小等を許容するものである。また、本願発明を用いて生産される装置、方法、ソフトウェア、システムが、その2次的生産品に登載されて商品化された場合であっても、本願発明の価値は何ら減ずるものではない。

産業上の利用可能性

[0188] 異なる分野に関連する広範囲のソフトウェアの問題を効率的に処理し、従来の方法論と比較した場合には、Lyeeを使用すると開発時間、保守時間およびドキュメント量はかなり少なくなる(70ー80%程度)。

図面の簡単な説明

[0189] [図1]本発明の一実施形態に係るセルの概念を説明するための概念図である。

[図2]本発明の一実施形態に係る相互に作用するプロセスの例を示す図である。

[図3]本発明の一実施形態に係る要件の実行の概念を説明するための概念図である。

[図4]本発明の一実施形態に係るLyeeパレットの概念を説明するための概念図である。

[図5]本発明の一実施形態に係る基本構造の概念を説明するための概念図である。

[図6]本発明の一実施形態に係る述語ベクトルの概念を説明するための概念図である。

[図7]本発明の一実施形態に係るL4_aおよびL4_bの述語ベクトルの概念を説明するための概念図である。

[図8]本発明の一実施形態に係るL3_aおよびL3_bの述語ベクトルの概念を説明するための概念図である。

[図9]本発明の一実施形態に係る画面相互作用の概念を説明するための概念図である。

[図10]本発明の一実施形態に係る処理経路図を説明するための概念図である。

[図11]本発明の一実施形態に係る画面が一つの場合を示す概念図である(実施例1)。

[図12]本発明の一実施形態に係るLyee計算法によるプログラム$P(S_1)$のプロセスとエンドユーザのプロセスを示すための説明図である。

[図13]本発明の一実施形態に係るプロセス間の相互作用を説明するための概念図である。

[図14]本発明の一実施形態に係る画面が二つの場合を示す概念図である(実施例2)。

[図15]本発明の一実施形態に係るLyee計算法によるプログラム$P(S_1)$のプロセスとエンドユーザのプロセスを示すための説明図である。

[図16]本発明の異なる一実施形態として、「開発対象のソフトウェア」を生産するためのプログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開

発支援装置、或いはソフトウェア開発管理装置のいずれかとして本発明を実施する場合に機能として備える構成を示した機能ブロック図である。

[図17]本発明の一実施形態に係る上記の構成を備えるプログラム(ソフトウェア)、プログラム生成装置、プログラム処理装置、ツール(装置として或いはソフトウェアとしての双方を含む)、ソフトウェア開発装置、ソフトウェア開発支援装置、或いはソフトウェア開発管理装置のいずれかとして実施される本発明の動作を示したフローチャートである。

符号の説明

[0190] W04 W04パレット

W02 W02パレット

W03 W03パレット

1601 全体制御部

1602 宣言情報挿入部

1603 制御関数Φ配置部

1604 制御関数Ψ配置部

1605 情報記憶部

請求の範囲

[1] 1つのプログラムとして実装するユーザ要件を、論理体ごとに、該論理体上の単語ごとに、単語名、定義式、該定義式の実行条件、入出力属性、単語の値の属性によって宣言(規定)する第1のステップと、

単語単位の宣言から、Lyee計算法による入出力チャネルをそなえたプロセス・セルとしてモジュール化された論理要素(L_2、L_3、L_4)および、作用要素(I_2、O_4、S_4)を作成する第2のステップと、

前記論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合に集合化する第3のステップと、

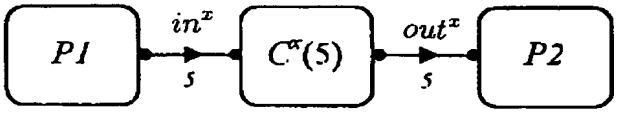
前記集合ごとに、1つの制御関数モジュールΦを配置する第4のステップと、

前記プログラムに1つの制御関数モジュールΨを配置する第5のステップと

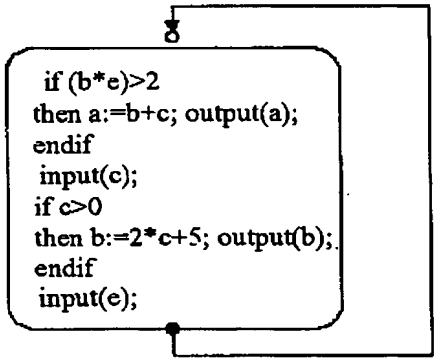
を具備することを特徴とするソフトウェア生成方法。

[図1]

[図2]

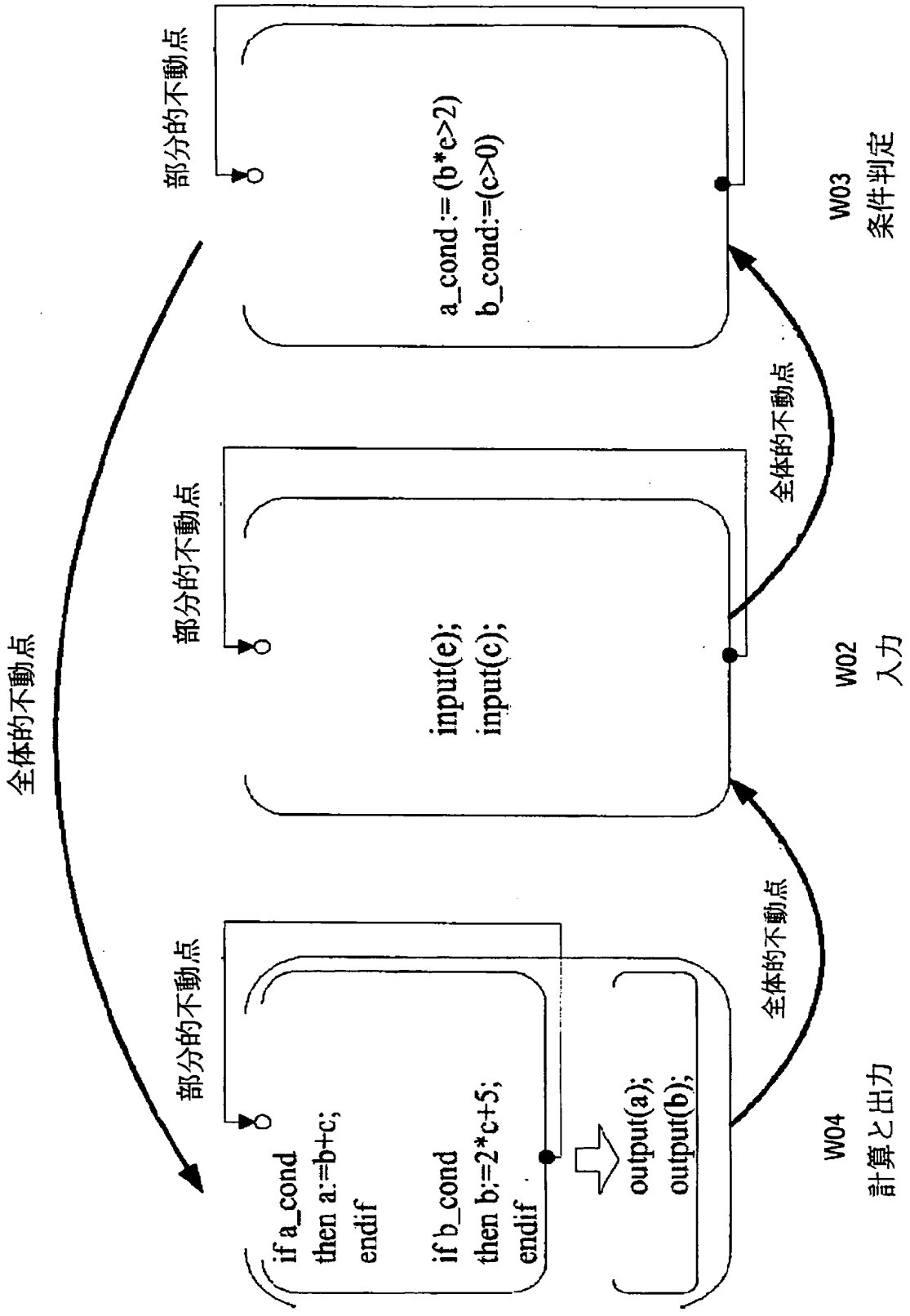


[図3]

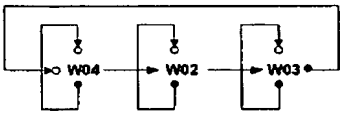


不動点に達するまで繰り返

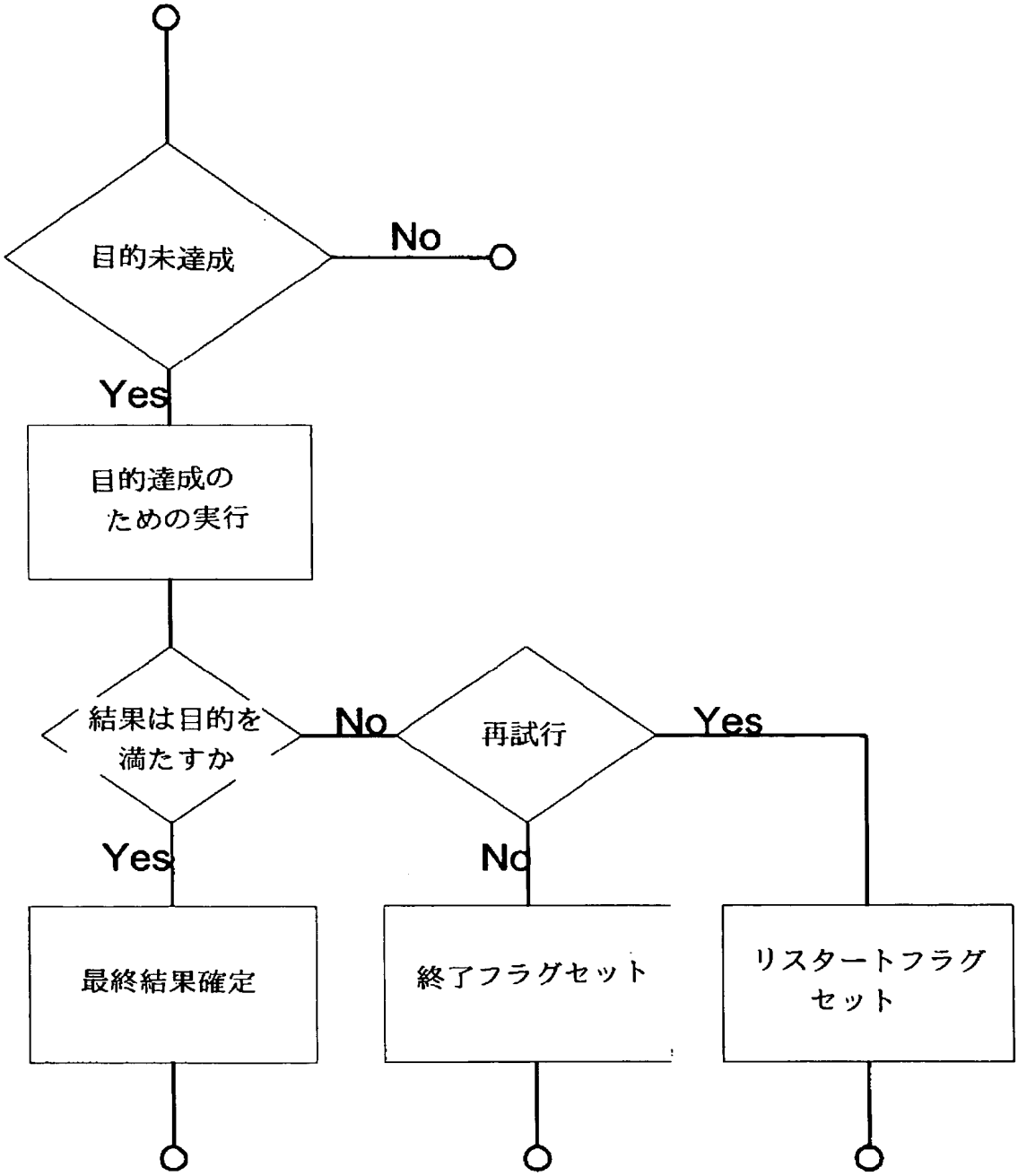
[図4]


全体的不動点
部分的不動点
if a_cond
then a:=b+c;
endif
if b_cond
then b:=2*c+5;
endif
output(a);
output(b);
W04
計算と出力
全体的不動点
部分的不動点
input(e);
input(c);
W02
入力
全体的不動点
部分的不動点
a_cond := (b*e>2)
b_cond:=(c>0)
W03
条件判定

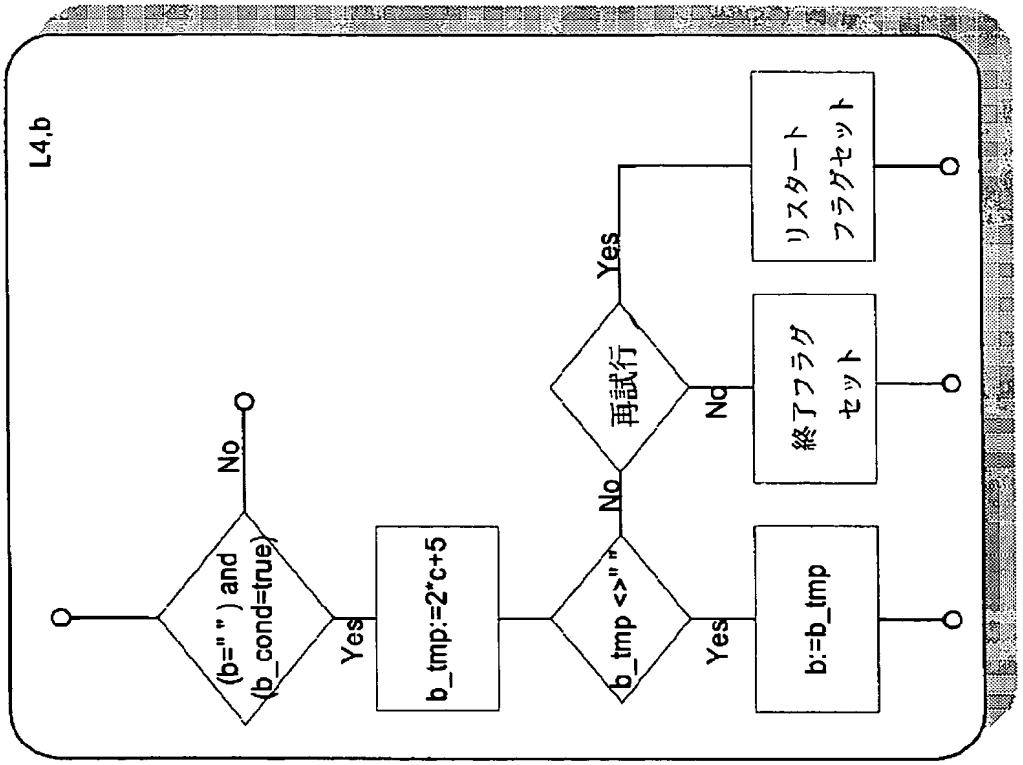
[図5]

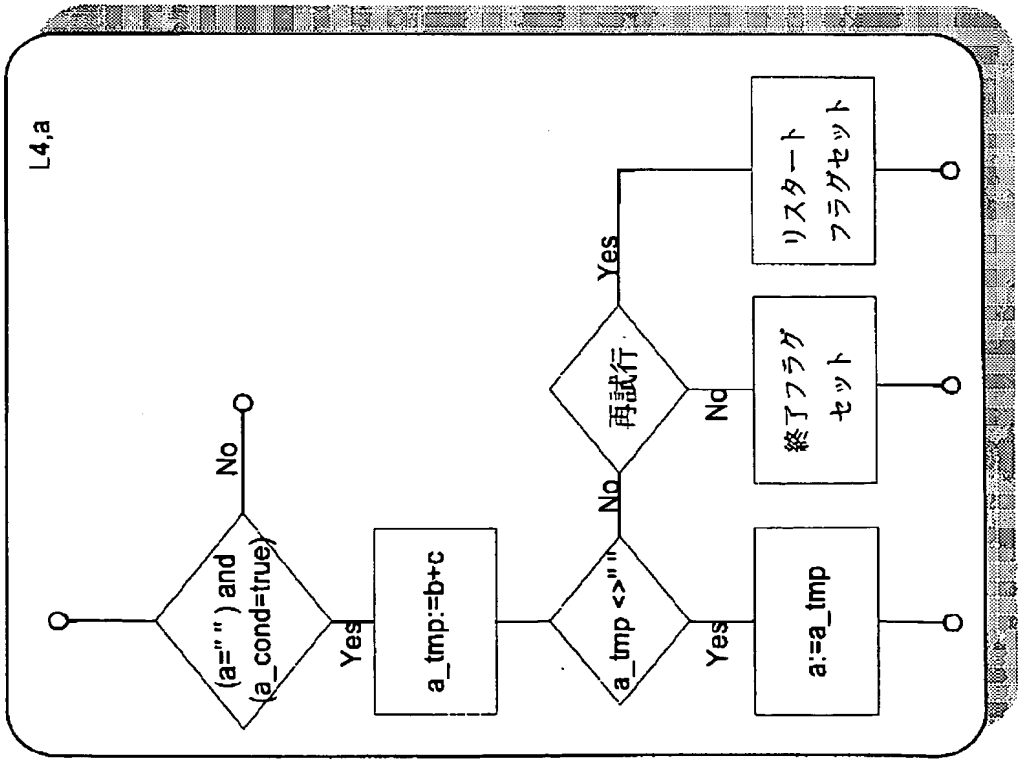

W04
W02
W03

[図6]

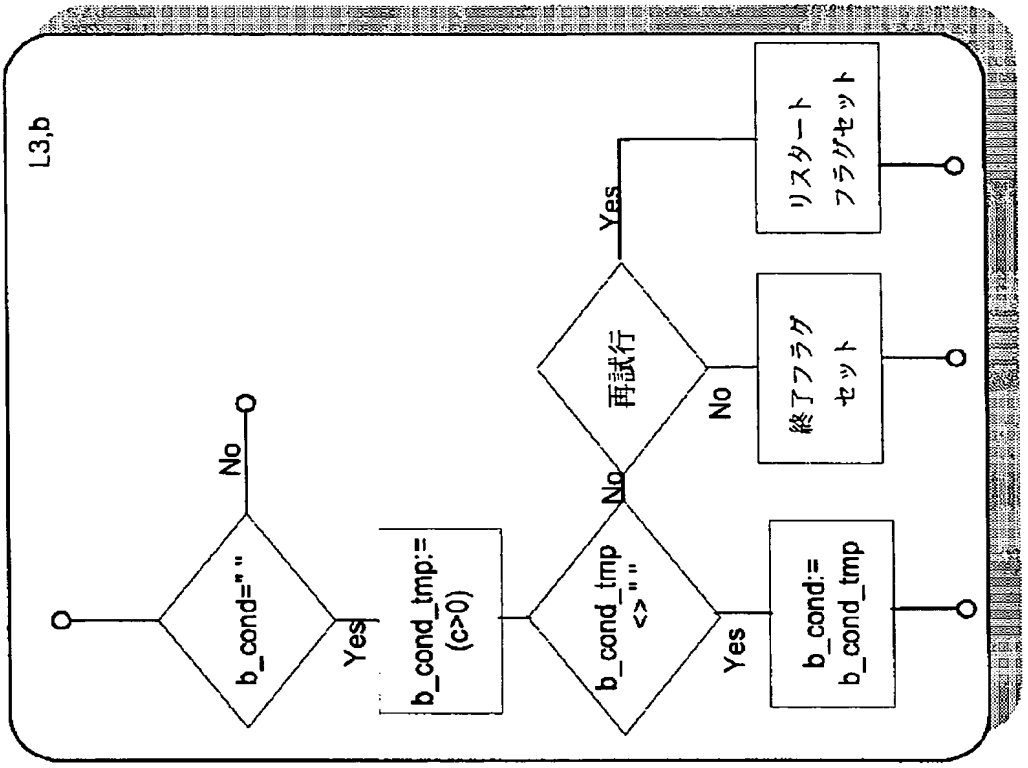

目的未達成
No
Yes
目的達成の
ための実行
結果は目的を
満たすか
No
再試行
Yes
Yes
No
最終結果確定
終了フラグセット
リスタートフラグ
セット

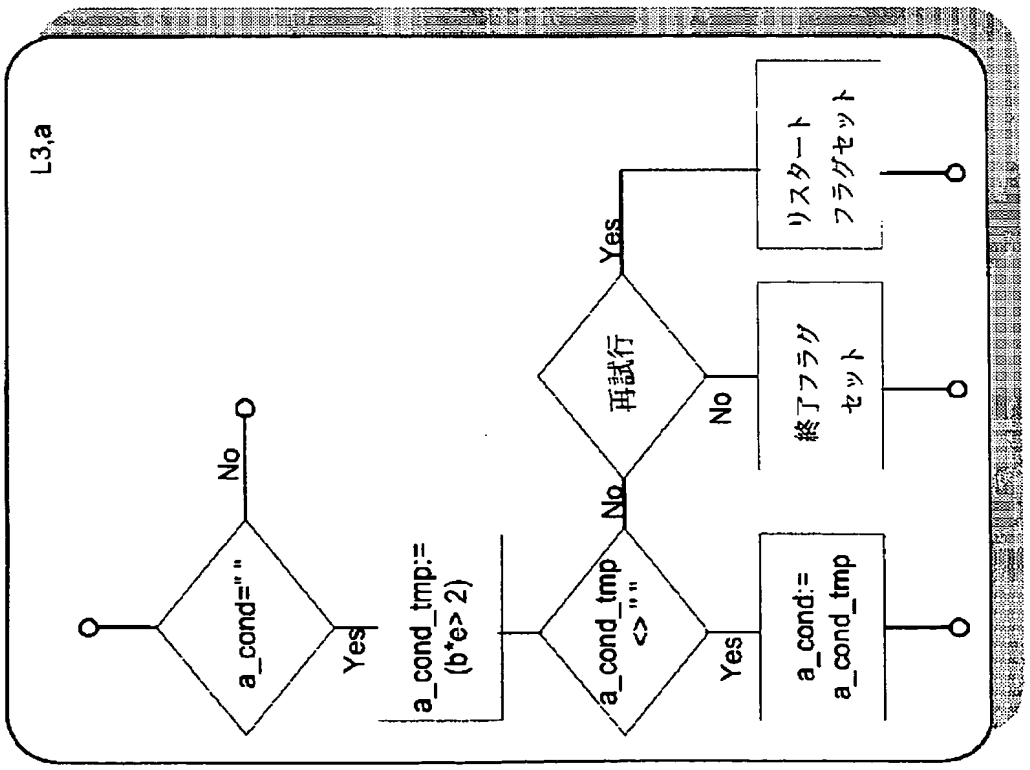
[図7]


L4,b
(b=" ") and (b_cond=true)
No
Yes
b_tmp:=2*c+5
b_tmp <>""
No
Yes
b:=b_tmp
再試行
Yes
No
終了フラグセット
リスタートフラグセット

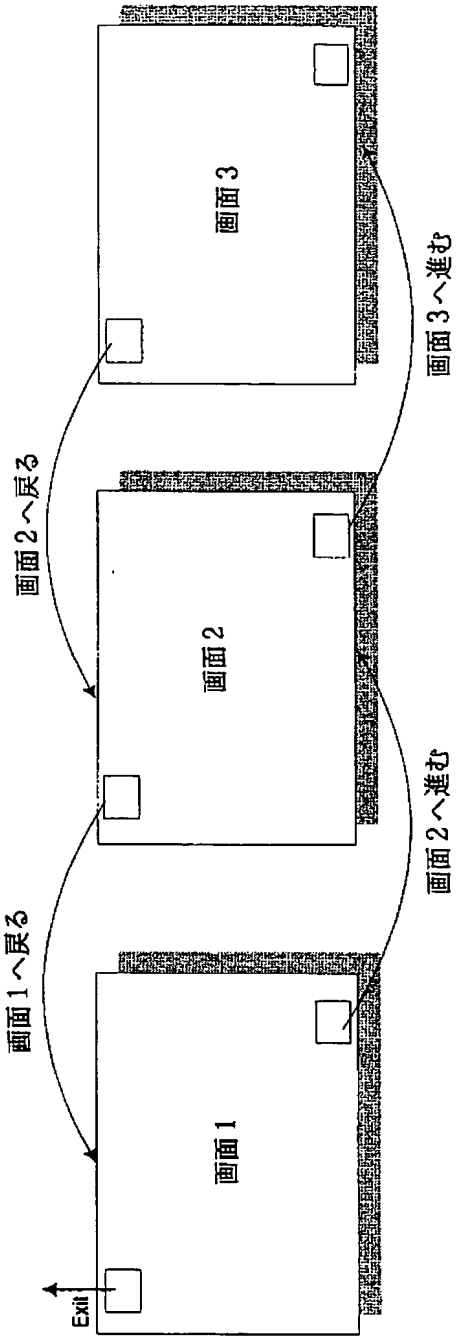

L4,a
(a=" ") and (a_cond=true)
No
Yes
a_tmp:=b+c
a_tmp <>""
No
Yes
a:=a_tmp
再試行
Yes
No
終了フラグセット
リスタートフラグセット

[図8]

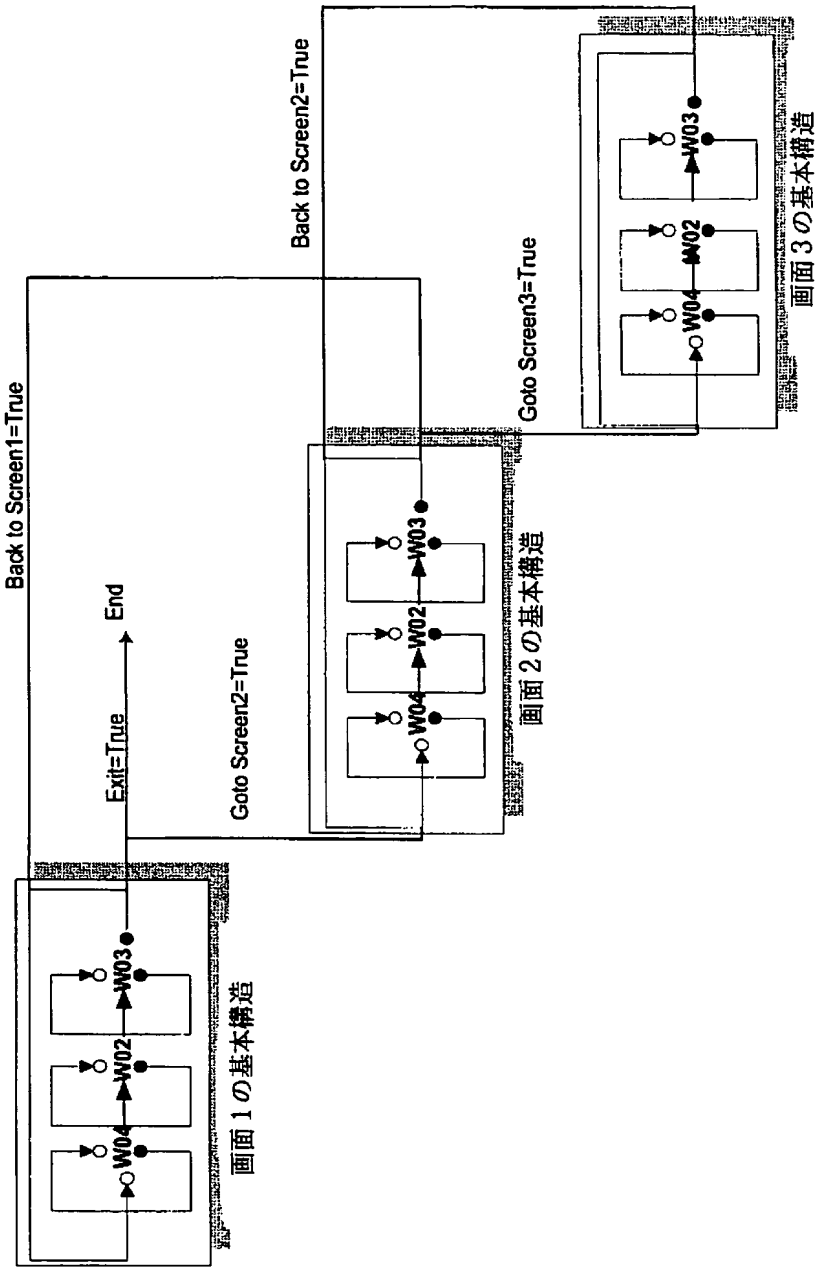

L3,b
b_cond=""
No
Yes
b_cond_tmp:=
(c>0)
b_cond_tmp
<>""
No
Yes
再試行
No
Yes
b_cond:=
b_cond_tmp
終了フラグ
セット
リスタート
フラグセット


L3,a
a_cond=""
No
Yes
a_cond_tmp:=
(b*e> 2)
a_cond_tmp
<>""
No
Yes
再試行
No
Yes
a_cond:=
a_cond_tmp
終了フラグ
セット
リスタート
フラグセット

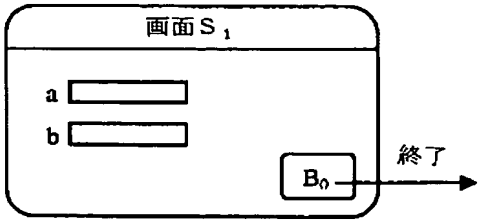
[図9]



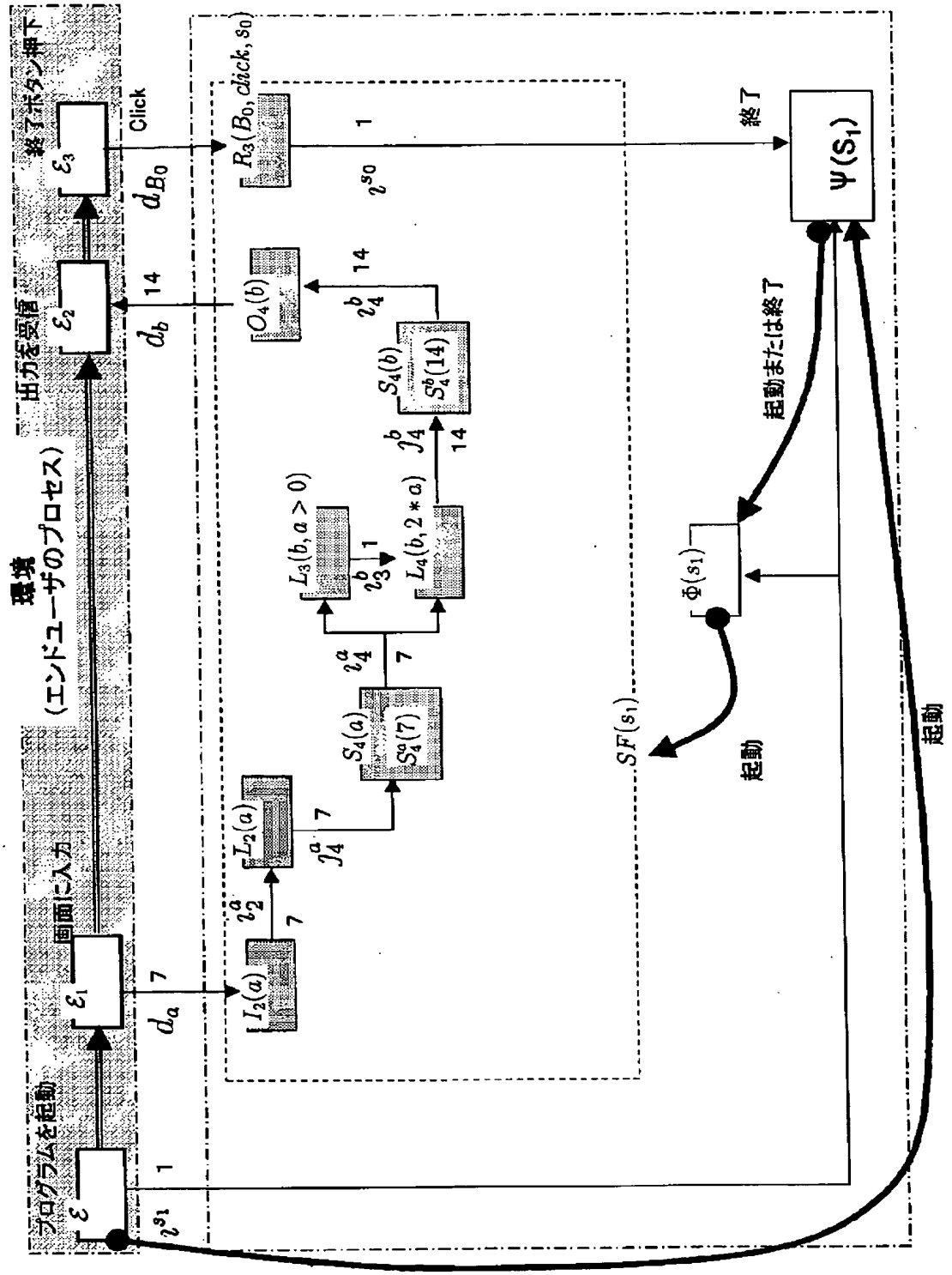
[図10]


Back to Screen2=True
Goto Screen3=True
Back to Screen1=True
End
Exit=True
Goto Screen2=True
W03
W02
W04
画面３の基本構造
画面２の基本構造
画面１の基本構造

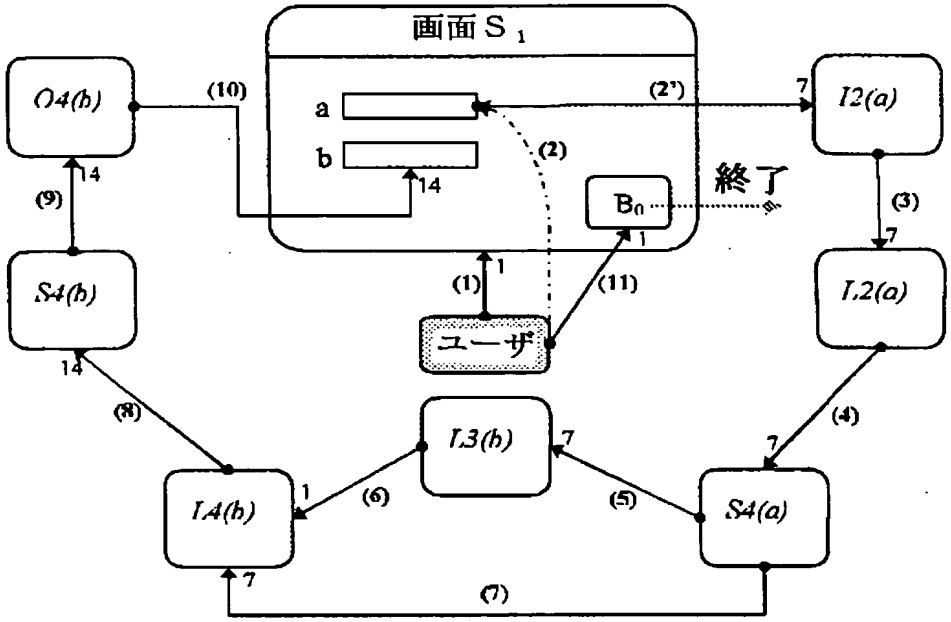
[図11]


画面S1
a
b
B0
終了

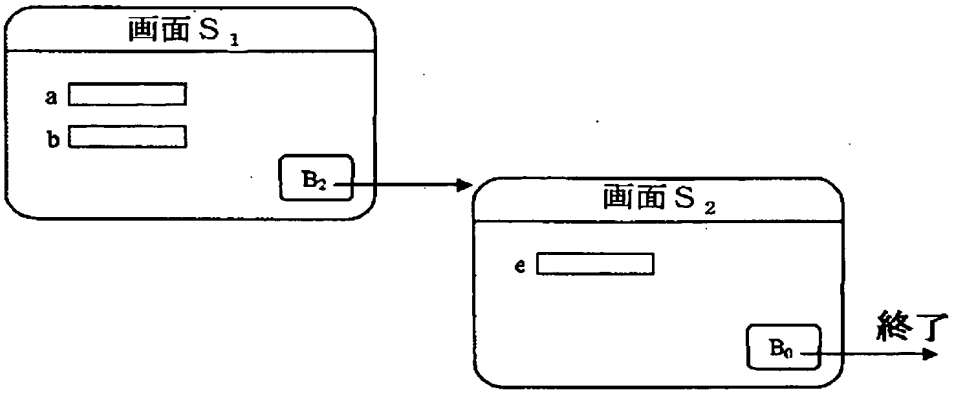
[図12]


環境
(エンドユーザのプロセス)
プログラムを起動
画面に入力
出力を受信
終了ボタン押下
Click
終了
起動または終了
起動
起動
$\Psi(S_1)$
$SF(S_1)$
$\Phi(s_1)$
$R_3(B_0, click, s_0)$
$O_4(b)$
$I_2(a)$
$L_2(a)$
$L_3(b, a>0)$
$L_4(b, 2*a)$
$S_4(a)$
$S_4(7)$
$S_4(b)$
$S_4(14)$
ε
ε_1
ε_2
ε_3
d_a
d_b
d_{B_0}
z^{s_1}
z^{s_0}
z_2^a
z_3^b
z_4^a
z_4^b
j_4^a
j_4^b
1
7
14

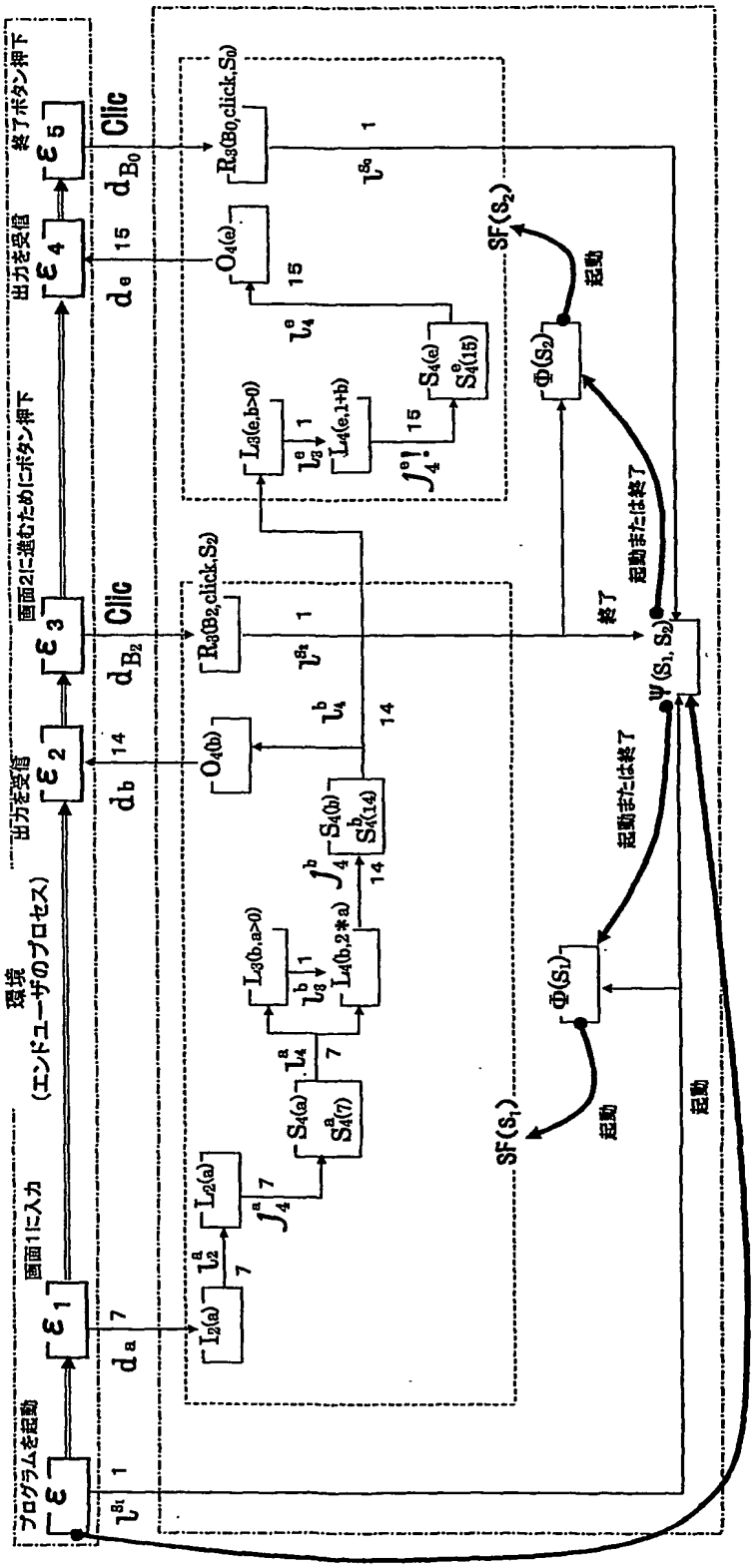
[図13]


画面 S 1
a
b
B0
終了
O4(h)
I2(a)
S4(h)
I.2(a)
ユーザ
I.3(h)
I.4(h)
S4(a)
(1)
(2)
(2')
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
1
7
14

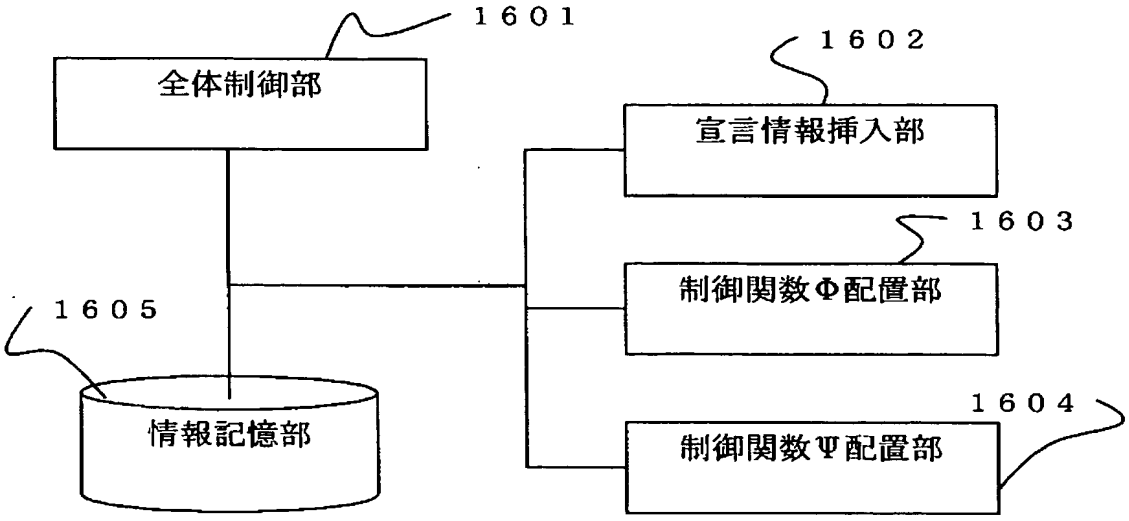
[図14]


画面 S 1
a
b
B2
画面 S 2
e
B0
終了

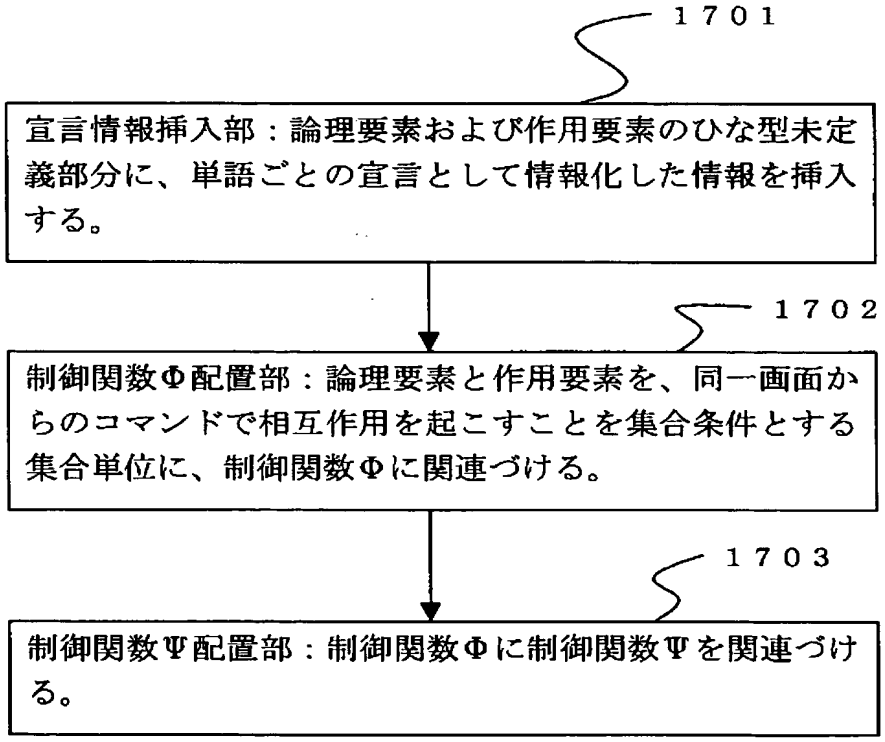
[図15]


プログラムを起動
画面1に入力
出力を受信
画面2に進むためにボタン押下
出力を受信
終了ボタン押下
環境
(エンドユーザのプロセス)
Clic
Clic
起動
起動
起動
起動
起動または終了
起動または終了
終了

[図16]


1601
全体制御部
1602
宣言情報挿入部
1603
制御関数Φ配置部
1605
情報記憶部
1604
制御関数Ψ配置部

[図17]


1701
宣言情報挿入部：論理要素および作用要素のひな型未定義部分に、単語ごとの宣言として情報化した情報を挿入する。
1702
制御関数Φ配置部：論理要素と作用要素を、同一画面からのコマンドで相互作用を起こすことを集合条件とする集合単位に、制御関数Φに関連づける。
1703
制御関数Ψ配置部：制御関数Φに制御関数Ψを関連づける。

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2004/013855

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F9/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Toroku Jitsuyo Shinan Koho	1994-2004
Kokai Jitsuyo Shinan Koho	1971-2004	Jitsuyo Shinan Toroku Koho	1996-2004

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	"Lyee Update <2003. January>", [online], 30 January, 2003 (30.01.03), Software Seisan Gijutsu Kenkyujo, [retrieval date 13 December, 2004 (13.12.04)], Internet <URL:http://www.lyee.co.jp/jp/lyeeupdate/docs/vol004lyeeupdate.PDF>	1
Y	WO 2001/035213 A1 (Information System Development Institute), 17 May, 2001 (17.05.01), Full text; all drawings & EP 1248189 A1 & CA 2359079 A & AU 1305401 A & NZ 512711 A	1

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 December, 2004 (15.12.04)	11 January, 2005 (11.01.05)
Name and mailing address of the ISA/ Japanese Patent Office	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2004/013855

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP 2001-5651 A (The Institute of Computer Based Software Methodology and Technology), 12 January, 2001 (12.01.01), Full text; all drawings & WO 2000/079385 A1 & CA 2414110 A & EP 1244006 A1	1
A	JP 2002-312167 A (Fujitsu Ltd.), 25 October, 2002 (25.10.02), Full text; all drawings (Family: none)	1

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２００４／０１３８５５

A.　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Ｉｎｔ．　Ｃｌ⁷　Ｇ０６Ｆ９／４４

B.　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Ｉｎｔ．　Ｃｌ⁷　Ｇ０６Ｆ９／４４

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－１９９４年
日本国登録実用新案公報	１９９４－１９９４年
日本国実用新案登録公報	１９９６－１９９４年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ｙ	”Ｌｙｅｅ　Ｕｐｄａｔｅ＜＜２００３．１月＞＞”，[online]，２００３．０１．３０，ソフトウェア生産技術研究所，[検索日　２００４．１２．１３]，インターネット＜URL:http://www.lyee.co.jp/jp/lyeeupdate/docs/vol004lyeeupdate.PDF＞	1
Ｙ	ＷＯ　２００１／０３５２１３　Ａ１（株式会社アイエスデー研究所）２００１．０５．１７，全文，全図　＆　ＥＰ　１２４８１８９　Ａ１　＆　ＣＡ　２３５９０７９　Ａ　＆　ＡＵ　１３０５４０１　Ａ　＆　ＮＺ　５１２７１１　Ａ	1

☒ Ｃ欄の続きにも文献が列挙されている。　☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日　１５．１２．２００４	国際調査報告の発送日　11.1.2005
国際調査機関の名称及びあて先 日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	特許庁審査官（権限のある職員）　５Ｂ　９３６６ 漆原　孝治 電話番号　０３－３５８１－１１０１　内線　３５４６

様式ＰＣＴ／ＩＳＡ／２１０（第２ページ）（２００４年１月）

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２００４／０１３８５５

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ｙ	ＪＰ　２００１－５６５１　Ａ（ソフトウェア生産技術研究所） ２００１．０１．１２，全文，全図　＆ ＷＯ　２０００／０７９３８５　Ａ１　＆ ＣＡ　２４１４１１０　Ａ　＆　ＥＰ　１２４４００６　Ａ１	１
Ａ	ＪＰ　２００２－３１２１６７　Ａ（富士通株式会社） ２００２．１０．２５，全文，全図（ファミリーなし）	１